ECMA

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

COMPUTER-SUPPORTED TELECOMMUNICATIONS APPLICATIONS

ECMA TR/52

June 1990

ECMA

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

COMPUTER-SUPPORTED TELECOMMUNICATIONS APPLICATIONS

ECMA TR/52

June 1990

BRIEF HISTORY

This Technical Report is the foundation for the OSI Layer 7 service Protocol Message Interface between Computing Functions and Switching Functions. The report introduces an Architectural and Operational Model to provide functional integration of applications using a computing network and private telephony network.

This Technical Report is based on the practical experience of ECMA member companies and results of their active and continuous participation in the work of ISO, CCITT, ETSI and various national Standardisation Bodies in Europe and North America.

Adopted as Technical Report TR/52 by the General Assembly of 28th June 1990.

Table of Contents

		Page
SECTION I - OVERVIEW		1
1.	SCOPE	1
2.	FIELD OF APPLICATION	1
3.	REFERENCES	2
4.	DEFINITIONS AND ACRONYMS	3
4.1	Agent	3
4.2	Application	3
4.3	Application Domain	3
4.4	Application Environment	3
4.5	Basic Call	3
4.6	Call	3
4.7	Complex Call	3
4.8	Computing Domain	3
4.9	Computing Function	4
4.10	Computing Sub-domain	4
4.11	Customer	4
4.12	Device	4
4.13	Directory Number	4
4.14	Event	4
4.15	Interconnection Service Boundary	4
4.16	Object	4
4.17	Party	4
4.18	Service	4
4.19	Service Boundary	4
4.20	State	4
4.21	Switching Function	4
4.22	Switching Domain	5
4.23	Switching Sub-domain	5
4.24	User	5
5.	INTRODUCTION	6
SECTION II - USES AND REQUIREMENTS		8
6.	APPLICATION EXAMPLES	8
6.1	Personal Telephone Support	9

6.2 Telemarketing 10

6.2.1 Outbound Calls 10
6.2.2 Inbound Calls 12
6.2.3 System Supervision 15

6.3 Customer Support Environment 15

6.3.1 Typical Application Modes 15
6.3.2 Typical Interactions 16

6.4 Integrated Message Desk 18
6.5 Emergency Call Applications 20

6.5.1 Industrial Alarm System 20
6.5.2 Fire Service Control 20

6.6 Data Collection/Distribution 21

6.6.1 General Scenarios 21
6.6.2 General Functions 21
6.6.3 Typical Examples 21

6.7 Data Access 23

6.7.1 Typical Application Modes 23
6.7.2 Typical Interactions 24

6.8 Hotel Application 24
6.9 Switched Data Applications 25

6.9.1 Computer Data Port Configuration 25

7. CSTA REQUIREMENTS 26

7.1 Standardization Requirements 27
7.2 System Requirements 27

7.2.1 System Integrity 27
7.2.2 System Reliability 28
7.2.3 System Independence 29
7.2.4 System Enhancements 29
7.2.5 Dispersion 30
7.2.6 Flexibility 30
7.2.7 Security 30
7.2.8 Compatibility 30
7.2.9 Distribution Transparency 30
7.2.10 Error Management 31
7.2.11 Performance 31
7.2.12 Telecommunications Configurations 32
7.2.13 Service Requirements 32

SECTION III - CSTA DESCRIPTIONS 33

8. FUNCTIONAL ARCHITECTURE 33

8.1 General Concepts 33

8.1.1 CSTA Application 33
8.1.2 Distribution of Computing and Switching Functions 35
8.1.3 CSTA Service 36
8.1.4 Client/Server Model 37
8.1.5 Service and Objects 37
8.1.6 Multi-Server/Multi-client Relationships 37
8.1.7 Application layer Relaying and Service Granularity 37
8.1.8 Service Boundaries 38

8.2 Distributed CSTA Functionality 39
8.3 Configuration Scenarios 40

8.3.1 Topologies 40
8.3.2 Reliability of CSTA configurations 42

9. OPERATIONAL MODELS 45

9.1 CSTA Domains and Sub-domains 45

9.1.1 Definitions 46

9.2 Switching sub-domain Model 47

9.2.1 CSTA Switching Model Objects 47

9.3 Computing sub-domain Model 55

9.3.1 Introduction 55
9.3.2 CSTA Computing Model Objects 55

9.4 Dynamic Identifier Management 56

10. CSTA SERVICE DESCRIPTIONS 57

10.1 Introduction 57

10.1.1 General Procedures 57
10.1.2 Categories of CSTA Services 59
10.1.3 Graphical Representation 59
10.1.4 CSTA Application Private Information 60
10.1.5 Other Parameters 60

10.2 Switching Function Services 60

10.2.1 Make_Call Service 60
10.2.2 Clear_Call Service 63
10.2.3 Clear_Connection Service 64
10.2.4 Hold_Call Service 65
10.2.5 Retrieve_Call Service 66
10.2.6 Consultation_Call Service 67
10.2.7 Alternate_Call Service 69
10.2.8 Reconnect_Call Service 71
10.2.9 Transfer_Call Service 72
10.2.10 Conference_Call Service 74
10.2.11 Answer_Call Service 76
10.2.12 Call_Completion Service 76
10.2.13 Make_Predictive_Call Service 77

10.2.14 Divert__Call Service 79
10.2.15 Feature Access Service 80

10.3 Input/Output Services 81

10.3.1 Collect__Data service 82
10.3.2 Data__Input Notification 84
10.3.3 Provide__Data Notification 85
10.3.4 Activate__Data__Path notification 86
10.3.5 Start__Data__Collection service 86

10.4 Computing System Services 88

10.4.1 Database Services 88

10.5 Bidirectional Services 92

10.5.1 Escape Service 92

10.6 Status Reporting Services 93

10.6.1 Snapshot Service 93
10.6.2 Monitor Service 94

11. CSTA STATES AND EVENTS 97

11.1 Introduction 97
11.2 Goals 97
11.3 Interpretation of call state 97
11.4 CSTA Telecommunications Scenarios 98

11.4.1 Making and Clearing a Call 99
11.4.2 Incoming Call 102
11.4.3 Holding Calls 103
11.4.4 Multiple Call Support 104
11.4.5 Tracking a Call 107
11.4.6 Events Prompted by the Remote Device 109
11.4.7 Queueing (inbound) 110
11.4.8 Predictive Dialling 111

11.5 CSTA Events 111

SECTION IV - CSTA APPLICATIONS 118

12. CSTA SERVICE BOUNDARY 118

12.1 Introduction 118
12.2 Service Decomposition Functional Elements 119
12.3 Location of Decomposition Functional Elements 119

13. APPLICATION LAYER STRUCTURE 121

13.1 Introduction 121
13.2 OSI Reference Model 121

13.2.1 General 121

13.2.2 Application Process 121
13.2.3 Application Entity 122
13.2.4 Application Service Elements 123
13.2.5 Application Associations 123
13.2.6 Application Context 123

13.3 CSTA Application Layer Structure 123

13.3.1 CSTA Application Process 124
13.3.2 CSTA Application Entities 124
13.3.3 CSTA Association and Context 124
13.3.4 CSTA Application Service Elements 124
13.3.5 CSTA Specification 124

13.4 Examples of common ASEs 124

14. INTERCONNECTION SERVICE ARCHITECTURE 125

14.1 General 125
14.2 Interconnection architecture modelling concepts 126

14.2.1 Convergence Functions 126
14.2.2 Subnetworks and their Interconnection 127

14.3 Criteria for the selection of lower layer protocols 128
14.4 Protocol stacks 129

14.4.1 OSI protocols 129
14.4.2 ISDN Interfaces 129

14.5 Consequences of the use of 'reduced' stacks 132
14.6 Management of Communication Functions 132

14.6.1 Reliability/redundancy 132
14.6.2 Congestion management - Recovery procedures 133

15. SECURITY AND MANAGEMENT IN CSTA 133

15.1 Security in CSTA 133

15.1.1 Overview 133
15.1.2 Authentication and access control 133
15.1.3 Security Policy 136

15.2 CSTA Management 135

15.2.1 Charging Information 136
15.2.2 Load Management 136

SECTION V - APPENDICES 137

APPENDIX A - REDUNDANCY 139

APPENDIX B - RELATING CSTA TO OSI LAYER 7 - ARCHITECTURE 143

APPENDIX C - VIRTUAL DEVICE CONCEPT 147

APPENDIX D - TRANSFER CONTEXT IN CSTA 151

APPENDIX E - OPEN DISTRIBUTED PROCESSING 153

SECTION I - OVERVIEW

1. SCOPE

This Technical Report defines an architectural framework suitable as a basis for the development of standards in the area of Computer-Supported Telecommunications Applications (CSTA). This edition of the report introduces a client-server Architectural Model and defines an Operational Model of the Switching Function acting as the server. This model is used as a basis for the definition of Switching Function Services.

This architectural framework is focused on the provision of an application interface between a Switching Function and a Computing Function. This application interface is not specified as being associated with a specific user-network interface or network-network interface. Hence the CSTA interface does not have to exist in the same physical interface as the objects with which it interacts. Therefore, CSTA is not intended to incorporate direct support for the user-to-network interface. Applications may incorporate support of user-to-network interfaces separately from CSTA but this is not within the scope of CSTA.

The definitions of an Operational Model for the Computing Function and the related Computing Function Services are provisional and will be completed in a future edition of the report.

Within the scope of the Operational Model embracing the Switching Function as server this edition of the report covers the following subject areas:

- types of CSTA application currently envisioned;
- overall system operational and functional requirements met within the framework;
- a functional architecture as it relates to a Telecommunications Application environment viewed as a whole;
- types of CSTA configurations envisaged;
- an operational architecture defined in terms of the Switching Function objects visible to, and capable of being acted on, by the Computing Functions;
- the individual Switching Function Services needed to support the CSTA applications envisaged;
- examples of some possible state transition scenarios as they relate to the Switching Function server interface to the Computing Function;
- the distribution of CSTA application functionality between the Switching and Computing Functions;
- Application Layer structure;
- the interconnection architectures existing to support the Application layer structure; and
- security and management within a distributed CSTA application domain.

2. FIELD OF APPLICATION

This Technical Report defines how to provide functional integration between a computing network and a private telecommunications network. The possibility of expansion is available, at a later stage, to cover public telecommunications networks.

The functions defined in this Technical Report allow for communication between the computing and switching networks to take place via intervening networks which range from a simple point-to-point connection to a local or wide area communications network.

In this Technical Report, the bias is towards:

- private networks,
- telephony services,
- uni-directional operations between computer and telecommunications networks with the telecommunications network as the server.

This Technical Report also introduces bi-directional services and non telephony services but these types of services have not been explored as extensively.

3. REFERENCES

ECMA

TR/44	An Architectural Framework for Private Networks
TR/45	Information Interchange for Remote Maintenance at the Interface between Data processing Equipment and Private Switching Networks
TR/46	Security in Open Systems - A Security Framework
TR/49	Support Environment for Open Distributed Processing

ISO

ISO 7498	Information Processing Systems - Open Systems Interconnection - Basic Reference Model
ISO 8648	Data Communications - Internal Organisation of the Network layer
ISO 8649	Open Systems Interconnection - Association Control Service Element
ISO 8824	Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1) (Note: This corresponds to CCITT X.208)
ISO 8825	Open Systems Interconnection - Specification of Basic Encoding Rules for ASN.1 (This International Standard corresponds to CCITT Rec. X.209)
ISO 10031-1	Telecommunications - Distributed Office Applications Model Part 1: General Model
ISO/DIS 9072/1	Text Processing - Remote Operations Part 1: Model, Notation and Service Definition
ISO/DIS 9072/2	Text Processing - remote Operations Part 2: Protocol Specification
ISO/DP 9545	Open System Interconnection - Application Layer Structure
ISO/DIS 10026/1	Open Systems Interconnection - Distributed Transaction Processing Part 1: Model
ISO/DIS 10026/2	Open Systems Interconnection - Distributed Transaction Processing Part 2: Service Definition
DIS 10026/3	Open Systems Interconnection - Distributed Transaction Processing Part 3: Protocol Specification

CCITT

Rec. I.210	Principles of telecommunications services supported by an ISDN and the means to describe them
Rec. Q.65	Stage 2 of the Method for the Characterization of Services Supported by an ISDN
Rec. I.130	Method for the Characterization of Telecommunication Services Supported by an ISDN and network Capabilities of an ISDN
Rec. Q.700	Introduction to CCITT Signalling System No. 7
Rec. Q.931	ISDN User-Network Interface Layer 3 Specification for Basic Call Control
Rec. Q.932	Generic Procedures for the Control of ISDN Supplementary Services
Rec. Q.940	ISDN User-Network Interface Management-General Aspects
Rec. X.25	Interface between DTE and DCE for Terminals operating in the packet mode and connection to public data networks by dedicated circuits.
Rec. X.75	Packet-Switched signalling between public networks providing data transmission services.

4. DEFINITIONS AND ACRONYMS

For the purpose of this Technical Report the following definitions apply.

CSTA Specific Definitions

The prefix CSTA applies to all terms included in the following list.

4.1 Agent

A CSTA user that is authorised to act on behalf of the provider of the CSTA application.

4.2 Application

A co-operative process between a Switching Function as performed within a telecommunications network and a Computing Function as performed within a computing network.

4.3 Application Domain

The union of one switching sub-domain and one computing sub-domain.

4.4 Application Environment

The set of application domains for an application.

4.5 Basic Call

A call that relates exactly two associated devices.

4.6 Call

A Switching Function communications relationship (generally) between two or more parties. During some circumstances, including set-up and release, there may be only one party.

4.7 Complex Call

A call that relates more than two associated devices.

4.8 Computing Domain

The set of computers and their objects which may be reached directly or indirectly by a CSTA application from a switching domain.

4.9 Computing Function

That part of the function needed to support CSTA applications implemented within a Computing Network.

4.10 Computing Sub-domain

Any configuration of inter-connected computers which present the external appearance and functionality of a single computer to the switching domain.

4.11 Customer

A person, process or piece of equipment that makes use of the services provided by an organisation on a regular basis.

4.12 Device

A logical entity which translates between the actions of a party and the (signalling) information transfer capabilities of the Switching Function.

4.13 Directory Number

A logical concept that translates to party or device. It is associated with a line circuit.

4.14 Event

A stimulus that causes a change in the state of a CSTA object.

4.15 Interconnection Service Boundary

The abstract service boundary within a system supporting a CSTA Application, separating the communications component of the application from the networking support functions of the system.

4.16 Object

An abstract entity assumed for modelling purposes to embody some aspect of the externally visible functional characteristics of a physical entity.

4.17 Party

An entity outside the Switching Function which has the intelligence to use the Switching Function.

4.18 Service

The benefit provided by one CSTA application process to another.

4.19 Service Boundary

The boundary existing between a CSTA Computing Function and a CSTA Switching Function as it is established via their Interconnection Service Boundaries over some underlying interconnection medium.

4.20 State

An indication of an object's current condition based on its passed events, permitting a prediction of its future behaviour.

4.21 Switching Function

That part of the function needed to support CSTA applications implemented within a telecommunications network.

4.22 Switching Domain

The set of all the switches and their objects which may be reached directly or indirectly by a CSTA application from a computing domain.

4.23 Switching Sub-domain

Any configuration of inter-connected switches which presents the external appearance and functionality of a single switch to the computing domain.

4.24 User

A person, process or piece of equipment that receives direct benefit (e.g. added functionality, improved performance) from the services provided by the CSTA application.

Terms Defined Elsewhere

ISO

ISO 7498	Application Layer Application-process Application-entity Application-entity-title Application-service-element
ISO 8648	Intermediate System Interworking Unit
ISO 8649	Application context Association Association Control service Element
ISO 9072/1	Remote Operations
ISO 10031-1	Client Server

List of Acronyms

ACK	Acknowledgment
ACSE	Association Control Service Element
ACD	Automatic Call Distribution
AE	Application Entity
AP	Application Process
APDU	Application Protocol Data Unit
API	Application Programming Interface
ANI	Automatic Number Identification
ASE	Application Service Element
ASN	Abstract Syntax Notation
CASB	CSTA Application Service Boundary
CO	Central Office
CSTA	Computer-Supported Telecommunications Applications
CF	Computing Function
CLI	Calling Line Identification
DDI	Direct Dial Inward
DND	Do Not Disturb
DNIS	Dialled Number Identification Service
DP	Data Processing
DTE	Data Terminal Equipment

DTMF	Dual Tone Multi-Frequency
FE	Functional Entity
FTAM	File Transfer Access and Management
HDLC	High-Level Data Link Control Procedure
ID	Identifier
ISDN	Integrated Services Digital Network
IWU	Inter-Working Unit
OSI	Open System Interconnection
PBX	Private Branch Exchange
POTS	Plain Old Telephone Service
SDFE	Service Decomposition Functional Element
PSTN	Public-Switched Telecommunications Network
PTT	Postal Telecommunications Telegraphics Administration
QoS	Quality of Service
RO	Remote Operation
ROSE	Remote Operation Service Element
SCCP	Signalling Connection Control Part
SF	Switching Function
SMDR	Station Message Detail Recording
SS7	Signalling System 7
TP	Transaction Processing

5. INTRODUCTION

This Technical Report provides an Architectural Framework within which to develop ECMA Standards in the area of Computer-Supported Telecommunications Applications (CSTA).

The basic aim of CSTA is the mutual enhancement of the individual capabilities of co-operating computing and switching networks to the extent that each may exploit the services provided by the other for the purpose of supporting applications that neither could provide individually without significant enhancement or redesign.

For example, a computer program running on a general purpose computer may make use of an attached voice switching network for the purpose of making and clearing calls in support of an integrated application involving both voice and data communication, an application unable to be supported by the computer alone.

Conversely, an application running on a switching network may make use of a database management system available on a connected computing network, an application unable to be supported by the switching network alone.

With respect to the second of these examples, it is recognised that most modern switching networks have their own database handling systems. For some applications, however, it may be advantageous to allow the network access to the more powerful and flexible database management systems available on a general purpose computer.

CSTA may be seen as representing a new area of Information Technology development for which Internationally agreed standards have yet to be defined. The purpose of this Technical Report is to establish an architectural framework within which to progress this work. The material making up the framework is presented in the following sequence:

Section II

Clause 6 deals in more detail with the types of application seen as being supported within a CSTA environment. It gives a number of application examples and illustrates for each the basic interactions between terminal, computer and telecommunications network components needed to support the application.

Clause 7 deals with the general operational, functional and engineering requirements to be covered by the framework. It also deals with the subject of conformance to the CSTA standards to be developed in accordance with this framework.

Section III

Clause 8 deals, under the heading of functional architecture, with basic concepts as they relate to a CSTA environment considered as a whole. It develops a model for the environment and identifies within the model the service boundaries seen as being the subject of CSTA standardization.

This clause also enlarges on the network configurations covered by the model.

Clause 9 defines, for each of the co-operating networks of the global model, an operational model. These models are defined in terms of the objects going to make up the networks as they are perceived and capable of being operated on by their companion networks.

On the basis of the models defined in clause 9, clause 10 defines the services provided by each network to the other and clause 11 the states and events relating to their service interfaces.

Section IV

Clause 12 considers in more detail the service boundary between co-operating computing and telecommunications network Application layer entities. It is concerned specifically with the distribution of CSTA Application layer functionality about the boundary and the way in which different distributions affect the definition of the service primitives exchanged over the boundary.

Clause 13 is concerned with defining the structure of CSTA Application layer entities in terms of their lesser functional components.

Clause 14 deals with the interconnection services required to support computer/telecommunication network interworking at the Application layer. It identifies the OSI and other lower layer protocol stacks able to provide the necessary interconnection services.

Clause 15 is concerned with security, including authentication and access control, within CSTA together with CSTA management.

Section V

Appendix A enlarges on the provisions of clause 8 dealing with interconnection scenarios.

Appendix B takes a practical CSTA Application example and illustrates how the provisions of the different clauses of the framework should be seen as being related.

Appendix C introduces the notion of a "virtual device" which may be an implementation mechanism for certain scenarios.

Appendix D introduces the concept of a Transfer Context service, which can be requested of the Computing Function by the Switching Function, to transfer the context of an application from one terminal to another.

Appendix E briefly summarises the concepts embodied in Open Distributed Processing and relates that work to the development of CSTA.

SECTION II - USES AND REQUIREMENTS

6. APPLICATION EXAMPLES

A CSTA application is a co-operative process between a telecommunications Switching Function and a Computing Function. It is presented to the user as a single application providing enhanced services. Typically the Computing Function may be distributed among a network of computers while the Switching Function may be distributed among a network of switches. The interactions between the computing and Switching Functions can be bi-directional such that the Computing Function provides services to the Switching Function and vice versa. In general, the user may be a human or some automatic entity.

The following application examples demonstrate the potential for CSTA. Although the list is not exhaustive, and does not address every category, it does identify the general types of interaction required. The applications described are as follows.

- Personal telephone support: provides improved human interface and support for users.
- Telemarketing:

  a. outbound calls: assists agents in making calls to customers

  b. inbound calls: assists agents in handling incoming calls from customers

  c. supervision: of the telemarketing activity
- Customer support environment: provides support for handling customer enquiries.
- Integrated message desk: integrates the telephony message desk facility with computer based electronic mail.
- Emergency call applications: including Industrial Alarm System and Fire Service Control examples.
- Data collection/distribution: exploits the voice terminal for the collection and distribution of data.
- Data access: supports the Switching Function accessing data held on a computer and vice versa.
- Hotel application: which identifies typical hotel telephony functions.
- Switched data application: provides support for configuration of data ports.

Typical examples of CSTA applications are described using a simplified configuration shown in Figure 1 where:

- a computer provides the Computing Function
- a switch provides the Switching Function
- a human CSTA user accesses CSTA enhanced services via a data terminal and a voice terminal
- the CSTA application uses a Public Switched Telephone Network (PSTN) to communicate with a remote user (termed customer in these examples).

Some examples of possible logical interactions are also provided. Although they indicate a potential distribution of function (among several possibilities) they do not imply that this is the optimum case. The logical interactions between the Computing and Switching Functions are contained in the CSTA Service Descriptions clause.

These examples refer to outbound (or outgoing) and inbound (or incoming) calls. These are with respect to the switch.

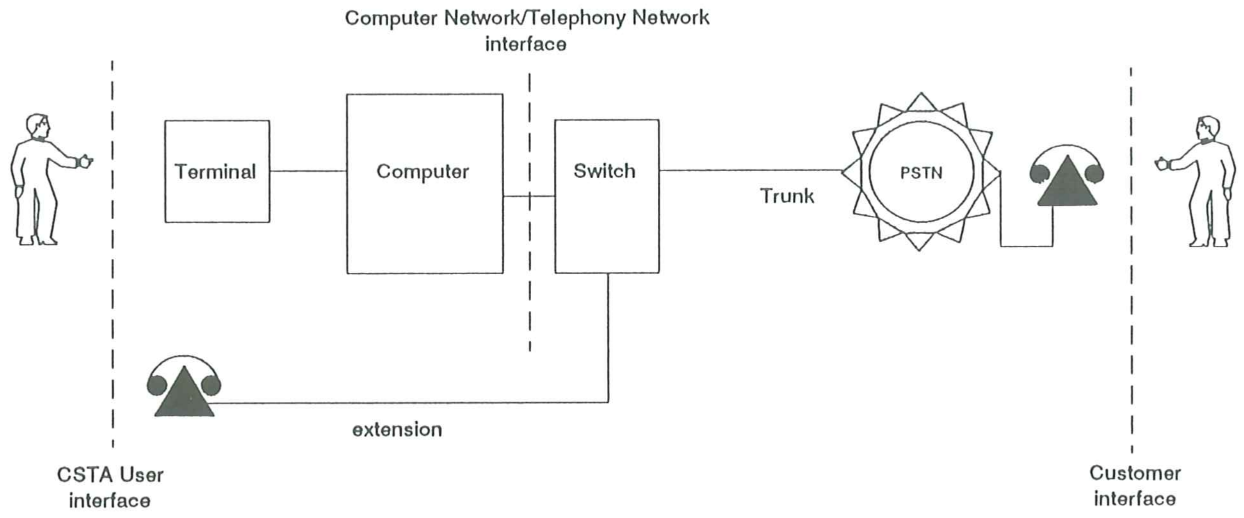


Figure 1 - Typical Simplified Configuration

6.1 Personal Telephone Support

This is concerned with providing the best human interface to users of the system while also monitoring user actions.

1. Users are presented with a simple, coherent and friendly screen-based interface to the system. Telephony functions, including call handling and feature invocation, can be invoked by the user interacting with his computer terminal. In addition to making the system easier to use, an application which presents a user-friendly interface will also reduce errors (such as mis-dialling etc.).

2. The application monitors the status of the individual user to provide improved co-ordination (including displaying call-related information on the data terminal). Examples of call status include:

   - detailed call progress states for outbound calls including indications that:

     - the call has been initiated

     - the call has been delivered (i.e. the far end is ringing)

     - the call has been answered (by the far end)

   - indications of failed calls (e.g. the called party is busy)

   - reports on changes to established calls (such as Call_Held or Call_Cleared)

   - incoming call indications (e.g. a call has arrived for a particular user)

   - status of the user's voice terminal (such as being in a do not disturb state)

3. The collection of adequate statistics on the activity and the performance of the system.

6.2 Telemarketing

This provides support for telemarketing agents (who become the CSTA user) in handling calls with customers. This type of application is normally telephony-intensive.

6.2.1 Outbound Calls

6.2.1.1 Typical Application Modes

Typically the application maintains a database of customer details (including contact telephone numbers, names, recent ordering activity, when next to call the customer, etc.) and uses this information to establish outbound calls. This operation can be done in various ways:

1. The simplest mode is where the application initiates a call, on behalf of an available agent, to the external number. Note that for simple telephones, the switch will normally prompt the local user to physically take his telephone off-hook; for more sophisticated voice terminals (such as hands-free voice terminals) this will not be needed. When appropriate the switch will initiate the outgoing external call. The extension user would typically hear call progress tones in the normal way. Usually the applications would also monitor the call progress by receiving event notifications as the call progresses through the various call establishment states. This call status information would typically be presented in an individual window on the computer terminal associated with the agent.

2. Some applications may have synchronisation requirements:

   - a request may only be valid under certain conditions. An example would be placing a new outbound call only for telemarketing agents in the idle condition (note that this is application dependent because there may be data terminal cleanup associated with the call becoming idle; in these cases, it may be best for the agent to explicitly indicate that he is ready for the next call);

   - it may be useful to defer a request such that it is only performed on the occurrence of certain events or at a certain time (an example would be to place a call every evening to report on the day's business).

3. Some applications prefer to use predictive dialling which requires access to detailed call progress indications. This mechanism initiates an outbound call without first assigning a telemarketing agent, but with the prediction that a suitable agent is likely to be available when the call is established. After initiating the call, the application monitors call progress and can detect call delivery (i.e. listening to ringback), call connection (i.e. the called party answers) and call failure (e.g. the called user is busy, number unobtainable etc.). At a suitable point within call establishment (depending on the application), the call is assigned by the application to an appropriate, available agent. Alternatively, if the application detects that the call has been unsuccessful (e.g. the called number was busy) then it records the information for a later attempt and moves on to the next customer with no interaction required from the agent. This type of sophisticated calling clearly requires extra functionality:

   - the ability to initiate a call that is not yet related to a particular extension;

   - access to detailed call progress signals;

   - the facility of assigning the completed (or partly completed call) to a free agent.

Although predictive dialling is more complex, the extra efficiency and the higher proportion of successful calls make it an important feature of telephony-intensive applications.

6.2.1.2 Typical Interactions

Typical logical interactions for an outbound call from an agent are shown in Figure 2. The particular interactions shown are intended only to indicate some possible types of exchanges; the actual interaction will, of course, depend on the application. Note that steps (1) and (2) are session initialisation and would normally only occur once for each individual agent session. The subsequent steps (3) to (12) are concerned with actually supporting outbound telephony and will normally occur once for each call involving the agent. Only the logical interactions between the computing and Switching Functions (i.e. steps 2, 3, 5, 8 and 11) are actually within the scope of CSTA.

Note that this example assumes call progress signals are provided by the network.

1. The agent logs on to the application. At this time, the application sets up the association between the user's voice terminal and his data terminal (this is typically operating system and configuration dependent).
2. The computer application, in this example, is interested in outgoing calls for this particular user. It requests notification of specified events associated with outgoing calls for the agent. Note that the control of which events are reported may also be effected by other means (such as administration).
3. The application requests the switch to initiate a call between the agent and the next customer in the list.
4. The network attempts to make the requested call between the agent and the customer.
5. The switch sends a report to the application indicating that the call has been delivered to the customer.
6. The application updates the relevant agent's data terminal with details of the called customer.
7. The called customer answers the ringing call.
8. The switch sends a report to the application indicating that the call has been answered by the customer.
9. The application now knows that the voice call (handled by the switch) is established.
10. The customer clears the call by hanging up.
11. The switch sends a report to the application indicating that the call has been cleared by the customer.
12. The application updates the agent's data terminal with full details of the completed transaction (including confirmation of order, length of telephone call, etc.).

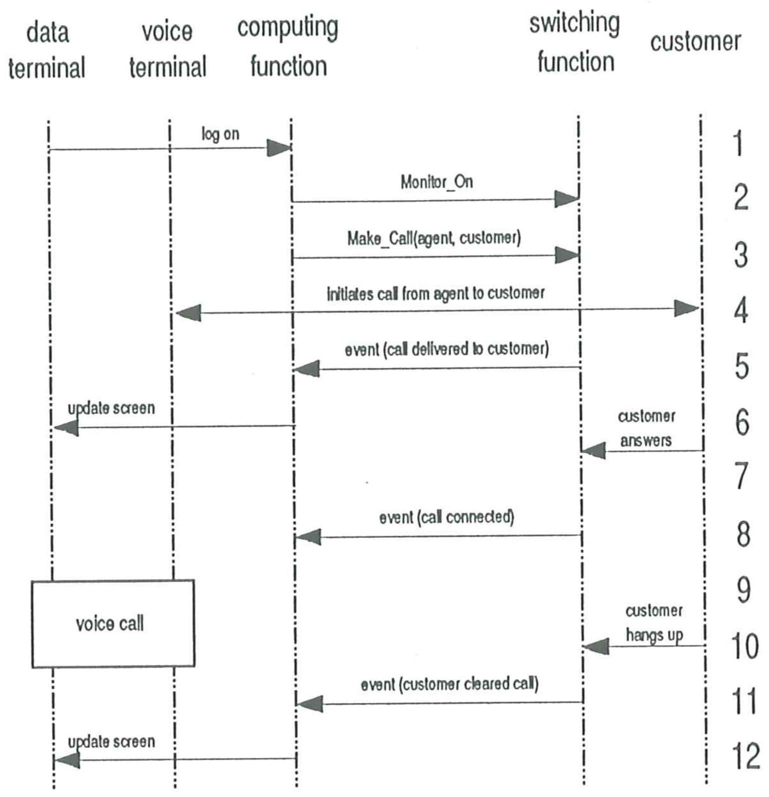


Figure 2 - Outbound Call Handling

6.2.2 Inbound Calls

6.2.2.1 Typical Application Modes

This type of application handles incoming calls and often exploits the call distribution functions offered by switches. Typical examples would include the use of Automatic Call Distributors (ACDs), based in the switch, to route an incoming call to the most suitable ACD agent out of a group of such agents. The application maintains a database of customer and/or product details which can be used to assist an agent in responding to incoming calls.

1. When an incoming call is received, the ACD application may make use of any information provided by the network on Calling and Called Line Identification to route the call to a particular agent.

   NOTE 1
   In some environments, the selection of the agent may be done by the computer using the Calling/Called Line information supplied by the switch, together with the monitored status of the individual agent.

2. When an incoming call is reported, applications may also make use of any information available on the calling or called number (if this type of service is provided by the network). People frequently make calls from places other than their normal

telephone so the calling number alone is often not sufficient to identify the actual calling person and in these cases, it may be necessary for the caller to identify himself by some other means. Examples of how calling and called number identification may be used are:

- to route the call to a particular agent and to simultaneously retrieve particular information from a database and display it on the agent's associated computer terminal;
- to prioritize calls such that VIP customers are placed at the front of the list and given to the first available agent (or even to an agent who specialises in important customers);
- to ascertain the reason for the call;
- to display the appropriate application screen in an environment where an agent supports multiple applications with each application having its own distinct pilot telephone number.

3. When an incoming call is delivered to a hands-free voice terminal (often used in telephony-intensive applications) the application might want to request that the switch answer the call on behalf of the agent.
4. On occasions, it may be that the call is delivered to an unmanned extension. In these circumstances, another agent would pickup the ringing call by interacting with his computer terminal.
5. It may also happen that the call is delivered to an agent who for some reason does not wish to accept the call. In this case, the agent can deflect the call to another specified agent by interacting with his computer terminal.
6. Under some conditions there may be no available agent when an incoming call is received. In this case, the call is redirected to an extension connected to a voice response unit. This operation will often be executed by the switch, but in some environments it may be the application that detects the call and requests the redirection. In either case, the voice response unit, under direction from the application, can respond to the caller and typically might record the caller's message for subsequent replay when an agent becomes free.

6.2.2.2 Typical Interactions

Typical interactions for an inbound call are shown in Figure 3. Only the interactions between the Computing and Switching Functions (ie steps 2, 4, 6 and 7) are within the scope of CSTA. The particular interactions shown are intended only to indicate some possible types of exchanges; the actual interaction will, of course, depend on the application. Note that steps (1) and (2) are session initialisation and would normally only occur once for each individual agent session. The steps (3) to (8) will normally occur once for each call involving the agent.

Note that this example assumes call progress signals are provided by the network.

1. The agent logs on and establishes an application session. The application sets up the association between the user's voice terminal and his data terminal (this is typically operating system and configuration dependent).
2. The computer application, in this example, is interested in incoming calls for this particular user. It requests notification of specified events for incoming calls for the agent. Note that the control of which events are reported may also be effected by other means (such as administration).

3. An incoming call indication is received by the switch which rings the agent's voice terminal.

4. The switch sends a report to the application indicating that a call has been received on this voice terminal. The switch will supply information about the call such as the Calling/Called Line Identification (if available) and whether the call had been forwarded, and if so, where from.

5. The application uses the supplied information to access a computer database and to update the called user's data terminal with the appropriate call details (e.g. caller, call forwarded details, etc.).

6. The application requests the switch to answer the agent's voice terminal. Alternatively the agent might physically answer the ringing call himself (in which case there is no direct synchronisation between the data terminal screen update (step 5) and the agent answering).

7. The switch sends a report to the application indicating that the incoming call has been answered.

8. The data session (handled by the computer) and the voice call (handled by the switch) are now established.

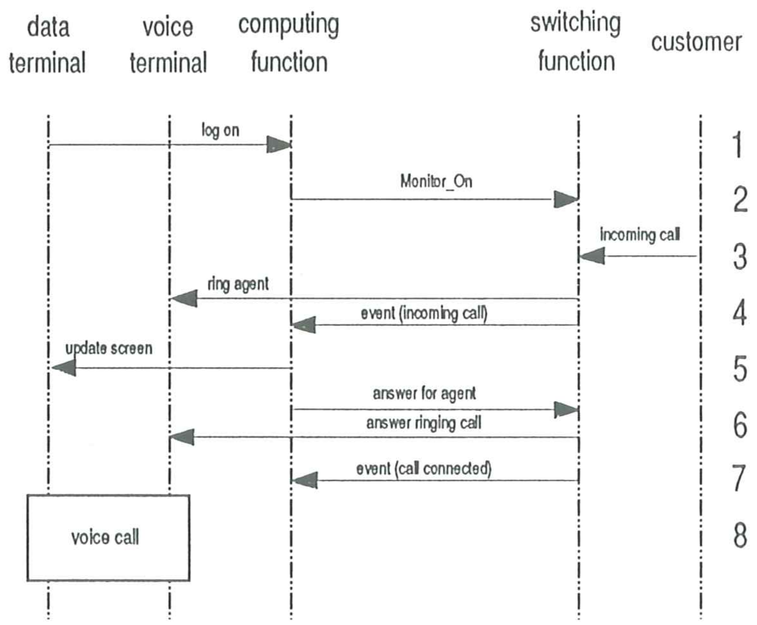


Figure 3 - Incoming Call Handling

6.2.3 System Supervision

Supervision is usually provided by sophisticated telecommunications applications. A supervisor would have a voice and a data terminal where the main interaction is via the computer terminal. Supervision can include:

1. Monitoring of each user and of their calls. A supervisor would want to be able to see individual actions and check their status via requests and displays on their computer terminal. As an example, agents sometimes simply go off-hook without placing a call as a means of rejecting any further calls and this type of action is of interest to the supervisor.
2. Monitoring of calls by type (so for example all calls to a particular pilot number are supervised irrespective of which agent is assigned to the call).
3. Many applications require the supervisor to participate in a call. This participation can be:
   - active: where the supervisor intrudes and talks;
   - passive: eavesdropping on the conversation by invoking some form of "silent" intrusion facility.
4. Silent intrusion (or conversation monitoring) is also useful when coupled with a digitised recording device. Many applications ask the customer if they may record the verbal transaction as verification of a sale or as a security check. In this case, a digitised recording devices is instructed to intrude into the connection and records the conversation along with pertinent customer information.
5. Statistics collection is also an important part of the overall application. Statistics collected (both on an individual customer and agent basis) include successful calls, unsuccessful calls and precise call durations.

6.3 Customer Support Environment

6.3.1 Typical Application Modes

This type of application is involved in handling customer enquiries. In many applications an agent is provided with an associated voice and data service. This is where the application provides a logical association between the agent's voice terminal call and his computer application session, such that the data displayed on his data terminal is related to his current telephone conversation. The application allows the agent to manipulate the two (voice and data) calls as though they were a single logical entity. A typical example is where an incoming call arrives for an agent together with the display of the customer's data on the computer terminal. During the conversation, this computer based customer record is updated by the agent interacting with his computer terminal. The typical types of operation that an agent would use in this scenario are:

1. placing the existing call on hold and making an enquiry call to a specialist. When the specialist answers the enquiry call, the application automatically displays the computer data on the specialist's data terminal;
2. retrieving the held combined transaction (voice and data) to resume conversation with the customer;
3. alternating between held and active transactions (where transaction includes voice and data);

4. it often happens that the agent will want to transfer the entire customer transaction (e.g. the voice call and the computer session) to a more appropriate agent (perhaps a specialist or supervisor);

5. clearing the customer's call (or any new enquiry calls) and simultaneously terminating the data session;

6. including additional agents into the transaction by forming a telephony conference where each conference member is also simultaneously presented with the details on their computer terminal screens. This means that the transaction details could be displayed on the data terminals of all the conference parties at the same time. Each conferee could independently scroll through pages of the transaction. This type of operation, as well as control over the database record and their update, would be application dependent.

Customer support environments are sometimes augmented by the inclusion of a voice response unit (providing voice functions such as speech recording, speech synthesis and speech recognition) as shown in the following figure.

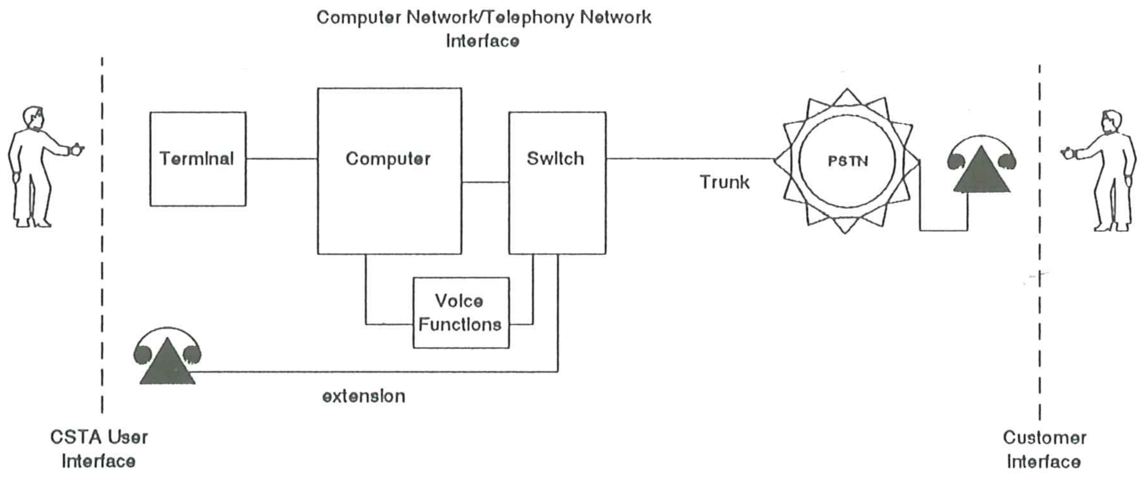


Figure 4 - Inclusion of a Voice Response Unit

6.3.2 Typical Interactions

Typical logical interactions for an application using a voice response unit are shown in the figure above. Only the interactions between the Computing and Switching Function (i.e. steps 2, 4, 7, 10, 11, 13, 15, 16 and 19) are actually within the scope of CSTA. This example covers an incoming call for a busy agent which is redirected to a voice response unit. This delivers an announcement followed by recording the caller's message. This message, together with associated information such as Calling/Called line identification, is stored by the application and the agent informed on his data terminal. When the agent becomes free, the application connects the agent's voice terminal to the voice response unit and replays the caller's message. The application then uses the recorded CLI data to place the return call to the original caller on behalf of the agent.

Note that this example assumes call progress signals are provided by the network.

Initialisation of the system is not shown but includes programming the switch to effect redirection of incoming calls when an agent is busy and the application requesting event notifications for the relevant extension lines.

1. The incoming call arrives at the switch for the (busy) agent.
2. The switch redirects the call to the voice messaging unit and notifies the application.
3. The application instructs the voice response unit to answer the call.
4. When the call becomes active, the switch notifies the application.
5. The application instructs the voice response unit to issue an announcement and record the caller's message. The application then retrieves and stores the message left by the caller.
6. The caller hangs up having left his message and expects an agent to call him back.
7. This call clearing by the customer is reported to the application.
8. The application informs the agent that a message is waiting for his attention (via his data terminal here, but it could be via his voice terminal if appropriate).
9. The agent clears his existing call.
10. The application is informed that the agent is now free.
11. The application requests the switch to establish a call from the agent to the voice response unit.
12. The switch attempts the Make_Call request by calling the specified agent and the voice messaging service.
13. The application is informed when this call is established.
14. The application instructs the voice messaging service to replay the message left by the caller.
15. When finished the application requests the switch to clear this call between the agent and the voice messaging service.
16. The application requests the switch to establish the return call from the agent to the original caller (using the stored Calling Line Identity).
17. While the call is being established, the application updates the agent's data terminal with data about the caller (retrieved from a database using the caller's identity).
18. The switch attempts to make the call from the agent to the customer.
19. The application is informed when the call is established between the agent and the original caller.

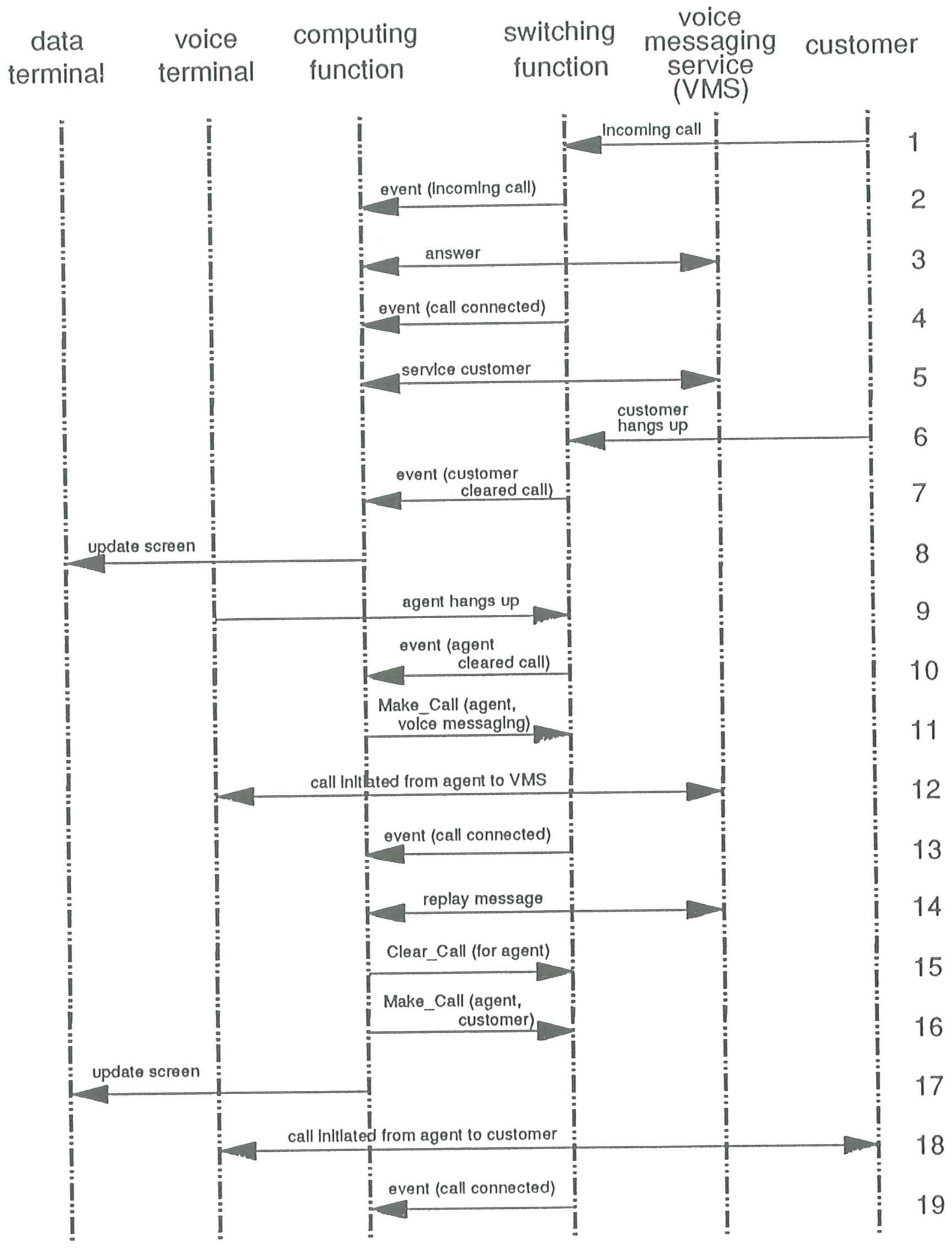


Figure 5 - Integrated Call Handling

6.4 Integrated Message Desk

Typically, a user invokes the service by providing information about projected absence (as described in Service invocation). Incoming calls are redirected to the appropriate attendant or answer point (as described in Message forwarding). The attendant handles the call on behalf of the called user as described in Attendant actions.

1. Service invocation:

   This can be via the user's voice or data terminal. The CSTA application conducts a dialogue with the user to ascertain such things as:

   - the user's schedule indicating where he will be at various times during the day;
   - the preferred message attendant to handle his calls;
   - if there are any callers that he would like to speak to urgently (rather than taking a message);
   - any particular messages he would like given to particular callers;
   - an emergency number (perhaps his manager's number).

   If the service was invoked via the voice terminal, then the switch effects the necessary call forwarding and informs the computer application about the details. If the service was invoked via the data terminal, then the computer requests the switch to effect the necessary call forwarding. In either case it may be necessary to ensure that the data held on the switch is consistent with that on the computer. Typically this would be done by one side (for example, the switch) requesting an update of all stored information held by the other side.

2. Message forwarding:

   This handles the redirection of incoming calls:

   - the application requests notification of incoming calls for the selected attendant(s);.
   - the switch call forwards the incoming call to the attendant and notifies the application:
     - that a call has arrived (i.e. is ringing on an identified attendant's voice terminal;
     - that the call has been forwarded from another extension (together with the number);
     - any calling or called line identification information that is available;
   - the application will use the supplied information (particularly the originally called extension number) to update the computer screen of the attendant with the appropriate data for the called user (i.e. the information originally entered by the user). Any data available about the caller would also be displayed.

3. Attendant actions:

   This will be based on the caller and the user supplied data (displayed on the attendant's screen). The attendant will often be attempting to make internal calls and would benefit from being able to invoke call completion features (such as recall when free) on encountering busy, or absent, users. The attendant's telephony operations (such as transferring the call) should ideally be invoked by the attendant interacting only with the computer terminal. Depending on the caller and the user's instructions the attendant might:

   - give the caller a specified message;

- take a message and forward to the user (for example via electronic mail including the caller's number in the mail header). The user could subsequently access items (in any order) from a single menu of electronic mail and telephone messages, giving the time received, the called number and the calling number. When a voice response unit is available, the application may be able to provide recording of the voice message as text, voice synthesis of the recorded text for playback, etc. The user may have the ability to place telephone calls directly from electronic mail messages generated by the attendant;
- if necessary, the user can be made aware of messages via the message waiting lamp on his voice terminal;
- transfer the incoming call to the user as indicated by the user's schedule;
- transfer the call to the supplied emergency number.

6.5 Emergency Call Applications

6.5.1 Industrial Alarm System

An industrial plant control computer system may be configured to include a capability to detect abnormal operation of the controlled plant and forward data to an alarm function. It is the purpose of the alarm function to analyse the data and, based on an exceptions database, take some action involving telephony communications.

In this case, we are concerned with an emergency scenario which requires a predetermined group of personnel to be contacted, the group being determined from the exceptions database dependent on the nature of the malfunction and other conditions, e.g. time of day. From this point, the following actions result:

1. A group of telephone numbers are retrieved from the database (a calling list).
2. Speech to be output to describe the emergency condition (e.g. type of fault, action within the plants, progress stage, etc.) is assembled. In the voice response unit this may be generated from text or segments of pre-recorded speech. Note that the voice response facility may be an option; live operator(s) are also possible and this would be analogous to "agents" in an outbound commercial telephony application.
3. The calling list and voice output being available, it is necessary to deliver the message in the minimum time. The use of CSTA offers an opportunity to minimise any delay. Some aspects for consideration are:
   - it may be necessary to invoke a particular outcalling algorithm (for example, those specified by PTTs) in order to differentiate emergency calls from routine calls;
   - it is desirable that emergency calls are given precedence, particularly when outgoing trunks are busy;
   - it may be desirable to override conditions such as "busy" or "do not disturb" at the called number.

6.5.2 Fire Service Control

A fire service Command and Control system may usefully employ CSTA principles to efficiently handle the inbound and outbound telephony activities concerned with a fire alert. A typical application scenario is:

1. Inbound: A fire alert is raised by dialling a fire emergency number with the call identified as an emergency call either by the called number (e.g. 999) or by operator intervention. The inbound call is delivered to the Command and Control console and

CSTA enables a pre-formatted data collection screen to be presented to the operator handling the call. Information to be collected includes caller's name and location, location and type of incident, etc. Ideally, the CSTA application should also be provided with the calling line identification allowing for some verification of the cited location and also permitting the identification of a suitable responding station. Automatic call recording is also invoked on all incoming emergency calls.

2. Outbound: When an incident has been reported and the location established, an optional, outbound "pre-alerting" call may be placed to all stations and units likely to be affected. Typically, these outbound calls are carried on leased lines in order to provide rapid call setup and to ensure availability for emergency applications. Fall-back to the PSTN is also provided. Having determined one or more responding stations, the incident data collected by the operator is transferred and a full alert raised.

6.6 Data Collection/Distribution

6.6.1 General Scenarios

This type of application can involve:

1. Using ordinary voice terminals to input (small) amounts of data for collection and analysis by a computer application.
2. Using special purpose devices (e.g. ID card reader) that sometimes contain a screen for output. These specialised terminals support a two-way dialogue between a user and the computer application:
   - the user enters information on the device which is conveyed, via the switch, to the application;
   - the application requests the switch to send output data to the device which is displayed on the screen.

6.6.2 General Functions

Various types of application are possible including Hotel: Room service; Hospital: Patient data entry; Wholesale: Ordering and inventory control; Administration: Identification and user guidance. Typical functions are:

1. Establish the connection.
2. Check access rights.
3. Transfer data to the computer.
4. Display data from the computer (including voice "data").
5. Park the "data collection/distribution" call to enable the use of the voice terminal for making another call.
6. Retrieve the "data collection/distribution" call and continue the data dialogue with the computer.
7. Release/cancel the connection (from user, switch or computer).

6.6.3 Typical Examples

6.6.3.1 Security Support System

A particular example of using ordinary, installed voice terminals to collect small amounts of input would be a Security Support System. This application is used to monitor a security guard as he patrols a building at night:

1. Certain voice terminals distributed within the premises are selected as input devices.
2. The application maintains a "geographical map" of these voice terminals and requests the switch to use them as input devices. All input entered at these voice terminals is collected by the switch and passed to the application. The patrolling guard uses each selected voice terminal in turn to report his progress (simply by using the voice terminal to dial an agreed number). Of course, extra sophistication can be easily added so that particular digits are used for defined messages (e.g. reporting on dirty offices, broken furniture, etc.).
3. The application logs the guard's report including location, time and message (if any).
4. The application monitors the guard's progress. It can:
    - automatically raise an alarm when an expected report is overdue (perhaps by requesting the switch to initiate a call from a voice response unit to the police station);
    - interpret the coded messages for display on a supervisory terminal;
    - keep track of the guard's whereabouts and automatically redirect any calls to the nearest voice terminal;
    - automatically log the fact that a complete patrol has been carried out successfully.
5. This application runs only at night and uses the switch in a particular mode. It would provide a more integrated solution if the application could also select particular classes of service for the switch (e.g. DAY and NIGHT service). This type of facility could also be generalised to allow access to other switch settings (for example, it would be helpful if the time and date maintained in the switch and the computer could be synchronised).

Although this example has the guard patrolling within a single building, there is no requirement for this restriction. The CSTA application should be capable of providing this service over standard interfaces to a public network.

If the voice terminal in this example had a display, then the guard could perform more complicated interactions such as obtaining new directions or reading computer supplied messages directly from the voice terminal. Manipulation of voice terminal displays would also be useful for other types of application (for example, hotel room service where a guest might interactively enter an order from a small voice terminal display).

The important common feature of these applications is that they could be implemented using only the voice terminal's signalling communications channel.

6.6.3.2 Voice Mail

Voice mail service integrated with a voice terminal that includes a display capability would allow the user to select voice mail functions based on visual rather than audio menuing. For example, a user might:

1. notice that their message waiting lamp was lit
2. press a voice mail softkey.
3. The CSTA application establishes:
    - a voice path (call) between the voice mail system and the user's voice terminal;

- a data path between the CSTA application and the user's voice terminal;
- a data path between itself and the voice mail system on behalf of the user.

4. The application visually presents a menu of options to the user such as "replay current message, listen to next message, listen to previous message, forward current message, record message, delete message", etc.
5. The user selects the desired menu options via the voice terminal's input facilities (e.g. dial pad, softkeys, touch screen, etc.).
6. When finished, the user exits the CSTA application.
7. The CSTA application disconnects the various paths established when the user invoked voice mail.

A similar application might allow the user to record a conversation on a voice mail system by selecting a "record" option from the display voice terminal.

The important common feature of these applications is that they require both voice and data paths to the voice terminal.

6.7 Data Access

6.7.1 Typical Application Modes

The switch and the computer can both support databases. The ability for one peer to access data held on the other peer could be exploited to develop integrated applications.

1. Synchronised directories: In Computer-Supported Telecommunications Applications it is likely that multiple directories will appear. The computer will, typically, maintain a database where some (or possibly all) of the same data is maintained. It is clearly preferable that the system manager only needs to update the system data at one place. In this example, we assume that the single management terminal is connected to the computer and is used for all administrative changes. The computer application maintains its own database and automatically sends requests to the switch to update the database held there. This could be extended to support "moves and changes" entered via a computer terminal to an application which generates the necessary requests for the switch. Examples of electronic directories, listed in hierarchical order from the terminal user's perspective are:
   - a personal directory of frequently called numbers;
   - a centrally maintained directory for the functional group (100 or so people) that one works with;
   - a directory for the corporate site where one works (a few thousand people);
   - the corporate directory made up from a concentration of all the site directories.
2. Directory access using dialled number: it is possible that a directory could be located on a computer and accessed, when required, by the switch. The switch would supply the computer application with a dialled number (which could in principle be inbound or outbound) and the computer application responds with the actual number to be used by the switch. A particular example, already in use, is support for "800" numbers. These are typically logical identifiers chosen to reflect some relevant aspect of the call (such as product range, geographical details, etc.) and a particular 800 type number is given to a customer who uses it as an ordinary telephone number when placing his call. This number can be used to ascertain information about the call and then converted to an actual telephone number.

3. Directory access using CLI: here the switch supplies the calling line identification, instead of the dialled number, to the computer in order to obtain the target directory number, i.e. the call routing is based on who is calling. For example, for emergency calls (e.g. 999 or 911) the call can be directed to the call handling centre nearest the caller's location. Another example is in departmental, or tenanting environments, where an operator call (e.g. 0), or message centre call, is directed to the person serving the department to which the caller belongs. A further example that would require this mechanism is in situations where a person only wishes to receive calls from a set of predesignated callers. All other calls are to be rejected or "deflected" to some alternate point.

6.7.2 Typical Interactions

Typical logical CSTA interactions to support the directory access examples are:

1. The switching system maintains a "trigger table" database which is checked when incoming calls arrive. Entries in the trigger table could be, for example, all calls to a particular telephone number or all calls from a particular trunk or set of trunks. This trigger table can be built up via switch administrative procedures or via dynamic registration procedures between the switch and computer.
2. When a "trigger" is activated, call processing is suspended and a request is sent to the computer with parameters such as CLI, a dialled number if any, and the associated "trigger point".
3. The computer can reply with instructions for the call to proceed and provide parameters such as target directory number, facility to be used and perhaps some identifying information to be supplied to the called party (e.g. caller's name). Alternatively, the computer may instruct the switch to reject the call and play an error tone.
4. The call proceeds as instructed.

6.8 Hotel Application

Typical hotel functions are supported by a "Front Office Computer" and a switch. Many of the functions can be performed independently but typical areas for integration are:

1. Room status modification: the maid dials room status changes using the room telephone and these can be passed on by the switch to the computer. Options are available to include identification of the maid together with monitoring of performance. Typically the maid might enter a code on entering the room (e.g. maid A, in room B, cleaning room) followed by a code when leaving (e.g. maid A, leaving room B, clean and empty, etc.).
2. Message waiting: the computer can request the switch to control the message waiting lamp on individual telephones.
3. Wake-up call: the computer can also manage the wake-up calls by requesting the switch to place the call to the designated telephone at the appropriate time.
4. Call accounting: in some environments it may be useful for the switch to send the computer call accounting information at the end of each call. Similarly, existing accounting information (for example, Station Message Detail Recording) can be sent from the switch to the computer.
5. Check-in/Check-out: these commands are entered from the front desk and sent to the switch to initialise data for the guest. This might include enabling/disabling the voice terminal, setting room meters to zero, passing invoice information, etc.

6. Directory: the computer can update the directory of the switch to reflect current occupation, etc.

7. Phone pre-payment: the handling of "pre-payment" for phones can be handled in the computer or the switch. In either case exchanges can take place to improve the operation of the system:

   - if the switch handles it, the computer can send the original amount (or modifications);

   - if the computer is handling it, the computer can send a request to disable the telephone when necessary.

8. Do not disturb: the computer can send requests to the switch to invoke these functions on behalf of guests.

9. Minibar/TV management: the initial input (such as entering consumption) is normally done via the room's telephone. This information can be passed to the computer for collation/action, etc.

6.9 Switched Data Applications

6.9.1 Computer Data Port Configuration

6.9.1.1 Background

Digital switches are commonly used to switch both voice and data. The data terminals are connected to the switch and then the switch to the computer ports.

Data terminals are connected to the switch using standard switch wiring. In this way a number of terminals can share access to the same computer ports resulting in port contention. The data terminal is usually connected to the switch via a data modem or direct interface. There are many types of data terminals and the switch would support a variety of Data Terminal Equipment (DTE) interfaces e.g. RS232, RS422, synchronous/asynchronous, etc.

To connect the switch to the computer data ports various types of interface are also available either RS232, RS422, Coax, etc.

6.9.1.2 Data Port Characteristics

Each type or model of data terminal has different characteristics and requires a different configuration of the computer port to which it is to be connected by the switch. To make the switched connection of data terminals more flexible it would be necessary to inform the computer, on a call-by-call basis, of the type of characteristics required at a given port.

6.9.1.3 CSTA Application Scenario

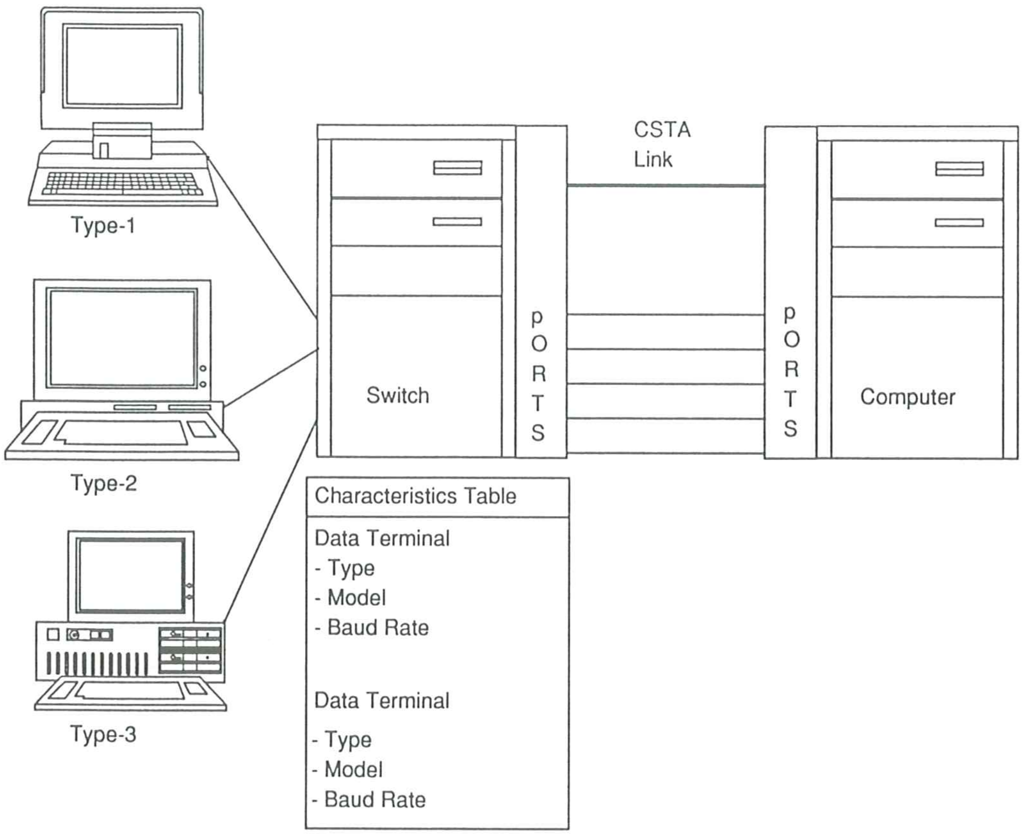


Figure 6 - Computer Data Port Configuration

The above diagram shows three data terminals of different types having access through the switch to data ports on the computer. The switch may determine some terminal characteristics automatically at the switch data interface (autobaud, parity etc.). Other terminal characteristics could be maintained in the switch by administration (type, model etc.). When a terminal initiates a data call, the switch will select the computer port. CSTA is used so that the switch can pass information on terminal characteristics and the port identifier to the computer so that the selected computer port can be configured appropriately. The data call can then go ahead.

With CSTA, data port configuration can be done on a call-by-call basis providing greater flexibility for switching data calls. The same service could also be used for selecting between various proprietary protocol emulations at the computer port.

7. CSTA REQUIREMENTS

Primarily the aim of the ECMA Interface Standard(s), which will eventually be written, is to define a list of common services together with interface protocol to assist computer and switch manufacturers in the support of CSTA. An additional aim is that the architectural structure of the interface will satisfy system performance and usage issues. The purpose of the CSTA Requirements clause is to set out the principles that will produce a robust, reliable and flexible interface.

These two aspects are covered below in two subclauses, the first covers the requirements on the standards and the second the overall system requirements.

7.1 Standardization Requirements

1. Scope: there is no attempt to standardise the applications. This means the standard(s) will define the services but not how they are used.

2. Independence: the CSTA services are implementation independent. That is the standard(s) are independent of the switch and computer type but also flexible enough to enhance existing telecommunications applications regardless of their implementation.

   NOTE 2
   There are different views on whether it is practical to strive for switching systems or computing systems presenting a common image across the switch to computer boundary.

3. Functionality: the integration of switch and computer within CSTA uses a peer-peer communication between an application process on the switch and an application process on the computer. This integration provides:

   a. Service Functionality independent of any particular application in the form of a set of services.

   b. Application Functionality which uses the services in a particular way.

      For a given CSTA application there will generally be several possible ways of splitting application functionality between switch and computer. In fact, the services imply a distribution of functions but it is beyond the scope of the CSTA standard(s) to define any split in application functionality. This is left to the discretion of the application developer.

4. Conformance: the purpose of the CSTA protocol standard(s) is to provide a list of standard services as introduced in clause 10. The structure of each service would be defined. There may be mandatory or optional information items and protocol procedures within each service. The question of defining platforms or collections of services and standardising those groupings is for further study.

7.2 System Requirements

The following considerations are important for the development of a robust and reliable interface. The standard list of services and any examples of their use will reflect these system requirements.

7.2.1 System Integrity

Telecommunication network users expect a level of service from network systems that must be maintained by any CSTA application. Requirements in this context are:

1. Normal audible call progress indications (e.g. dial tone, recorded announcement, music on hold etc.) must still be available.
2. CSTA should not add any avoidable delay to call handling procedures.
3. There should be a reliable fall-back action in the event of system failures.
4. Normal telephony should not be precluded by a failure of the CSTA system.
5. Partial CSTA service should still be possible under some failure conditions.
6. CSTA message flow should, as a general principle, be kept to a minimum.
7. A load control mechanism between the CSTA functions will be required.

Central offices and trunk types vary by PTT and locale. Trunk supervision signals depend upon several network dependent factors such as whether the trunk is digital or analogue, whether the network is fully or partly digital, the type of call, how the network sets up and clears calls, the type of DDI trunk signalling and so on. Certain applications (such as predictive outbound telemarketing requiring trunk supervision) would not be possible on some networks. This suggests the following requirements:

1. CSTA call models should consider the different network signalling types that will be encountered.
2. It should be stated which CSTA services rely upon trunk supervision.
3. It should be stated which CSTA services might be used in applications subject to PTT regulations.

7.2.2 System Reliability

CSTA systems should be robust, dependable and fault tolerant. The perceived reliability of telephone systems should not be affected by the presence of a CSTA application. In particular, CSTA should allow designs that include duplication in order to withstand individual failures. Distributing the server and client has particular implications for system reliability:

1. Independent failures can mean that one part of the distributed application can fail leaving the surviving part unsure of the precise status (e.g. whether a requested operation was performed or not, etc.). This introduces a requirement for an agreed recovery mechanism to cater for such independent failures (for example, tidying up incomplete service requests, etc.).
2. Partial overload can arise when one part of the distributed application becomes overloaded. This introduces a requirement for:
   - a technique for other parts of the application to detect this situation;
   - some form of backoff mechanism to allow the overloaded part to recover.
3. Communication failures can be hidden but prolonged, or total failures (such as network partitioning,) will be visible to the applications.

   This introduces a requirement for agreed procedures for when communications between the client and server is no longer possible.
4. Compatibility problems (e.g. version control) are exacerbated when the system is distributed. This introduces a requirement for compatibility checking.
5. Error management related to errors particular to distributed systems is required. There are various execution semantics to be considered. In decreasing implementation difficulty these are:
   - remote execution occurs exactly once;
   - remote execution occurs at most once including none or incomplete;
   - remote execution occurs at least once (including multiple incomplete and complete operations).

   CSTA should allow choices a. (e.g. for requesting functions) and b. (e.g. for reporting events).

Reliability means ensuring operation of the application and services to users. The CSTA design should account for the fact that the components within the system may have different levels of reliability. The details of redundancy, for example, are discussed in Appendix A, but several conclusions can be drawn from an analysis of the basic computer-switch system. In the event of a switch failure the reliability factor is solely dependent upon the redundancy in the switch - in many private switches and all central office switches this capacity is a pre-requisite. In the event of the computer, or the communications infrastructure failing, there are two stages to default functionality on the switch. Firstly, the operation of call handling functions must remain intact; secondly, the operation of the telephone sets must not be affected. With these considerations built into the CSTA standard, the service to users will be best maintained. These considerations suggest the following requirements:

1. Redundancy issues should be resolved from a system viewpoint.
2. Default functionality for applications must maintain switch service to users.
3. The status of the CSTA system should be checkable from both the computer and the switch.
4. In the event of a failure of the CSTA communications:
    - computer terminals should still operate
    - terminals in the switching network should still operate
    - normal telecommunications should still be possible.

7.2.3 System Independence

The CSTA architecture must be independent of the design of the switch and computer. In particular, it should allow the development of portable applications. A single application would clearly execute in a single piece of equipment. When it becomes distributed, the definition of the interactions between the parts must allow for heterogeneous environments (e.g. different languages, operating systems, data representations, etc.). To solve the problems introduced by these different environments it is required that the remote interactions are specified in an implementation-independent form. This requires the use of a formal interface specification language and an agreed external data representation which are independent of the choice of programming language, operating system, computer, switch and networks.

The CSTA architecture should also be independent of:

1. The network topologies.
2. The mechanism used to implement the requested service.
3. The type of network transport mechanism.
4. The type of service user.
5. The system ownership.
6. The interface to the public telephony network (e.g. PSTN trunk type).

7.2.4 System Enhancements

In order to achieve the required level of co-operation between the computer and switching networks, it will be necessary to enhance the operational facilities of both networks. In this context the term "operational facilities" is taken to include not just the traditional operating system functions (scheduling, I/O handling, etc.) but also the various utilities and applications (e.g. terminal handlers, database management systems, etc.) which are "layered" on top of the operating system and are available for use by the CSTA Application. Two categories of enhancement will be required:

1. Internal: each network may be required to perform new functions, usually invoked as a result of requests from the other network. For example, the switching network will

need a new function to provide monitoring of extensions as a result of a Monitor service request from the computing network.

2. External: each network will require the ability to form and send service requests and responses to the other network and to receive and interpret them in return.

7.2.5 Dispersion

A CSTA application process must be able to control and monitor terminals on widely distributed networks, not just on the machine on which it is located.

7.2.6 Flexibility

The architecture for CSTA should be flexible and extendable. In particular, it must be sufficiently flexible to support both Plain Old Telephone Service (POTS) and ISDN terminals. It must also be capable of supporting different network types (e.g. voice networks, data networks). The architecture must be extendable to allow for timely introduction of new applications. It must be unrestricting, allowing a wide breadth of application solutions.

7.2.7 Security

CSTA will approach the notion of security as outlined in ECMA TR/46 (Security in Open Systems). In this case, "security" will refer to characteristics of distributed applications that give resistance to accidents, failure and misuse, intentional or otherwise:

1. Protection of access: this controls access to defined resources for certain known users and includes:
   - authentication which is concerned with proving the user's identity;
   - access authorisation which is concerned with granting access to protect the confidentiality and integrity of security objects. Various methods may be used to authorise access (such as access-control lists, capabilities and other security attributes, singly or in combination).
2. Protection of information: interchanged or stored within distributed applications from external attack.
3. Protection of usage of resources: this is concerned with the usage of resources including secrecy and preventing denial of service.
4. Accountability of usage of resources: this includes selective logging of an audit trail of operations.
5. Management: this provides tools to manage the security facilities.

7.2.8 Compatibility

The CSTA architecture must be compatible, where appropriate, with related standards, including CCITT Recommendations for telecommunications and ISO International Standards for computing.

7.2.9 Distribution Transparency

The client/server model allows for a single application to be distributed via a "services definition". A major issue in this distribution of function is whether or not to hide distributedness and its consequences. There are arguments for (primarily simplicity for the user) and against (mainly performance and implementation complexity) the provision of full transparency. Different transparency requirements are likely and CSTA should not preclude implementation choices (for example, it would be possible to present transparency but allow visibility of distributedness via optional parameters, etc.). Particular aspects of transparency to be considered are:

1. The distribution may be via an intervening network of any level of complexity (e.g. from a simple point to point link to a local or wide area network).
2. There should be an option of automatic recovery attempts from any failure of the underlying connection.
3. Since different parts of the application are executing in different environments, there are particular problems associated with one of the parts becoming overloaded. It should be possible, therefore, for such conditions to be reflected in the definition of the interactions.
4. Support of multiple client/server relationships (i.e. by multiplexing) over a single underlying connection.
5. It should be possible to provide redundancy in the connections supporting the client/server such that if one connection becomes unavailable another can be used. This also gives the possibility of load sharing.
6. The distribution mechanism should not impose any operational constraints on the interface (i.e. either partner must be able to request a function at any time).
7. The distribution mechanism should deliver requests in the order they were issued.

7.2.10 Error Management

When dealing with a distributed application it is possible to distinguish between:

1. Execution errors which are related to the execution of the function itself. These errors are independent of the distribution of the parts and include the following categories:
   - normal: denoting successful operation;
   - warning: denoting that the operation completed but detected a situation of interest to the invoker (e.g. parameter ignored, end-of-file encountered);
   - abnormal: denoting a serious error which prevented successful completion. Not all output may be returned but termination is orderly (e.g. missing output is indicated);
   - error: denoting execution failure and no output (e.g. invalid input parameters).
2. Environmental errors which are related to the distributed nature of the system. Examples are:
   - remote system crash;
   - unrecoverable communications problems;
   - naming and binding problems (e.g. the remote application cannot be located, is currently not available, etc.);
   - security problems (e.g. the client is not allowed access to the requested service);
   - protocol or syntax problems (e.g. invalid syntax, invalid request, non-supported feature, etc.).

   These errors must be handled in the definition of the interaction between the distributed parts of the application.

7.2.11 Performance

The CSTA architecture must be such that the performance of applications can be specified in terms of switch and computer real time efficiency.

7.2.12 Telecommunications Configurations

In order to support a wide range of application types in a flexible manner, a range of telecommunications configurations and resources need to be supported in CSTA. Different application scenarios will require different configuration arrangements and resources. For example, a simple screen assisted telephony application may request CSTA services on behalf of a particular extension whereas a sophisticated outbound telemarketing application may require visibility of a set of agents, calls and potentially of a set of trunks as well.

Telecommunications resources of interest to CSTA applications can be physical interfaces (e.g. lines, trunks, etc.) or logical constructs such as agent groups, inbound call queues or trunk groups. The types of telecommunications resources of interest to CSTA includes:

1. Simple voice terminals where calls can be originated, answered or manipulated. These devices are typically identified by a unique directory number and are capable of handling one or more related calls (e.g. an active call and a call on hold).

2. Multi-line sets are terminals with many "lines" or "buttons" each addressable by a directory number and equipped with a call indicator (e.g. LED, LCD or lamp). These sets can thus handle, or have access to, many different calls simultaneously.

   Multi-line sets can be configured in various ways. In the simple case, each line "appearance" or "button" has a unique directory number. In the more complex case, the sets are arranged in work groups where a given directory number has "appearances" on more than one set (e.g. key systems). Thus, there is a "one to many" relationship between directory numbers and physical sets (i.e. a line appearance cannot be uniquely identified by a directory number). CSTA applications may need to address individual line appearances. This will then allow CSTA to originate, or answer, calls at a particular physical set while preserving the inherent flexibility offered by multi- appearance directory numbers.

3. Trunks are the interface points to other switching domains in the network; they are also used for connectivity within the domain. For incoming calls, the identity of the trunk is useful in determining where in the network the call has originated, or the kind of service requested. A telemarketing application could use the trunk identity to route a call to the appropriate agent queue, or allocate appropriate priorities. For outbound calls, the application may wish to select the trunk. To cater for these applications, when trunks are an access to another domain they need to have an identifier or number available to the CSTA application.

4. Groups such as hunt groups or agent groups are call distribution mechanisms. Typically when all agents are busy, queueing is supported. Telemarketing applications will require visibility of groups, available agents, and associated queues to flexibly manage a call handling operation. Groups are typically identified by a directory number.

5. Calls are associations between a set of CSTA devices. They are the primary service of the Switching Function, and usually have costs associated with the amount of time they exist. Because they are a convenient way to refer to a group of devices that interact with one another, and because they can be deflected, routed, queued and terminated, they are of interest to CSTA applications.

7.2.13 Service Requirements

There are two general types of interaction required between the functions:

1. The ability to ascertain what is happening in the remote function (an example of this type of interaction is shown via event reporting in clause 6).

2. The ability to request services to be performed by the remote function (an example of this type of interaction is also shown in clause 6).

SECTION III - CSTA DESCRIPTIONS

8. FUNCTIONAL ARCHITECTURE

8.1 General Concepts

This clause introduces the concept of a client/server relationship as it applies generally to CSTA Application functions performed within co-operating computing and switching networks, irrespective of the direction in which the relationship is established.

8.1.1 CSTA Application

A CSTA application is defined as a co-operative process between a telecommunications network Switching Function (SF) and a Computing Function (CF).

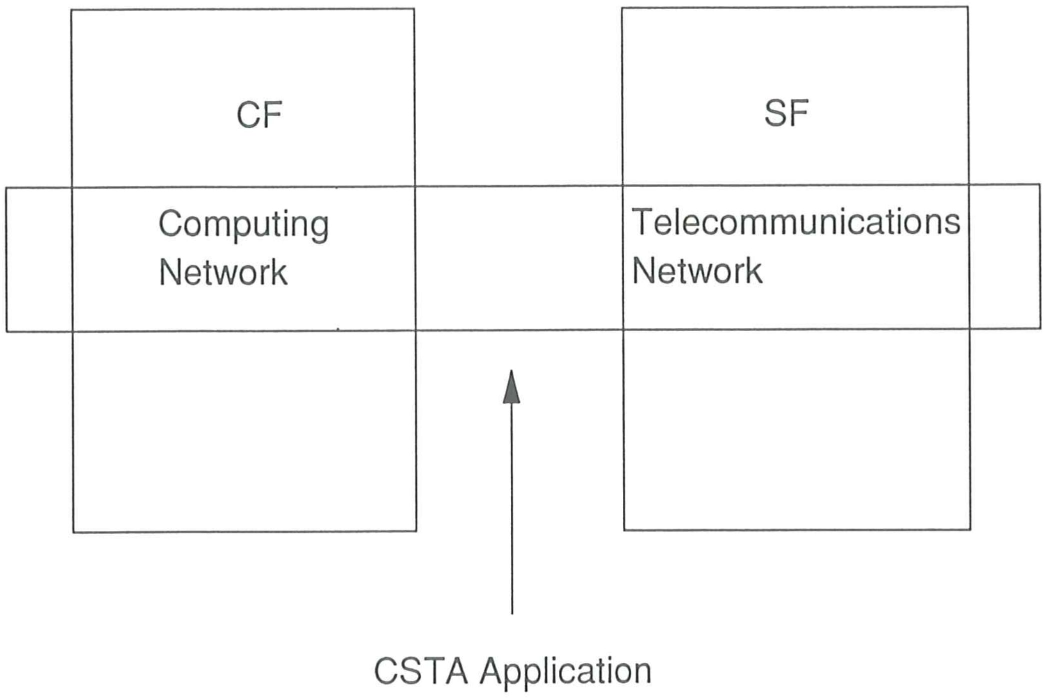


Figure 7 - Co-operation between CF and SF

The objective of the proposed CSTA Architecture is to define the interworking mechanisms between these functions in a way which is independent of their physical implementation.

The CSTA application will be supported by a computing component (normally based in the computing network) and a switching component (normally based in the telecommunications network). The operation of these components is defined by one or more interactions between them as shown in Figure 8.

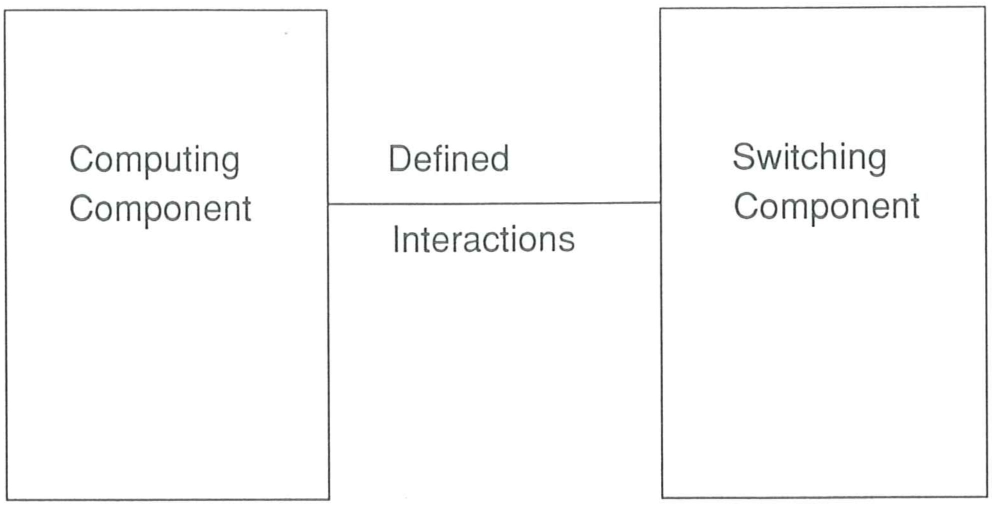


Figure 8 - Interaction between Computing and Switching Components

8.1.2 Distribution of Computing and Switching Functions

Typically, but not necessarily, most of the Computing Functions are implemented by one or several computers in a computing network, while the Switching Functions are implemented by one or several switches in a telecommunications network. It is, however, possible for some Computing Functions to be performed within the telephony network and some Switching Functions within the computing network:

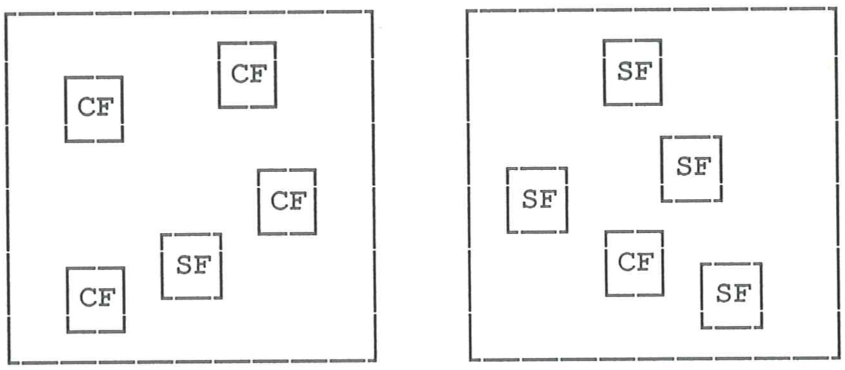


Figure 9 - Distribution of Computing and Switching Functions

All Switching Functions performed within the computing network are regarded as being outside the scope of this edition of the Technical Report.

In practice, CSTA is designed to allow "users" of either computing or telecommunications networks to have access to an enhanced set of functions which will be provided by the other network. Thus a computer "user" who was previously able to make use of the facilities available on a conventional computer, is now able to use programs which make phone calls and perform other telecommunications functions. Conversely, a "user" of the telecommunications network is now able to make use of facilities available on the computer, such as access to database systems of greater power and flexibility than those on the telecommunications network.

In both cases a "user" can be human, but it is also possible for the term to relate to an automatic entity within one or other network. For instance, a screen based Directory program or a telemarketing package driven (largely) by a human user. On the other hand, in an automatic call distribution package (a system for interpreting incoming telephony addresses to decide to which telephone to deliver a call) the "user" is an application program within the switching network.

The CSTA system will appear to the user, human or machine, as a single application on a single network, not as two separate applications on two separate networks (as it will, in fact, be implemented). This fact is expressed by referring to this single application as the CSTA application.

Since the components of the CSTA applications will (in most situations) be distributed, some form of communications support is required. This can be shown by expanding each of the components (identified earlier) into application functionality (to support the defined interactions), CSTA services (to support the necessary exchange of messages) and lower layer interconnection service provider. The relationship is shown in the following figure.

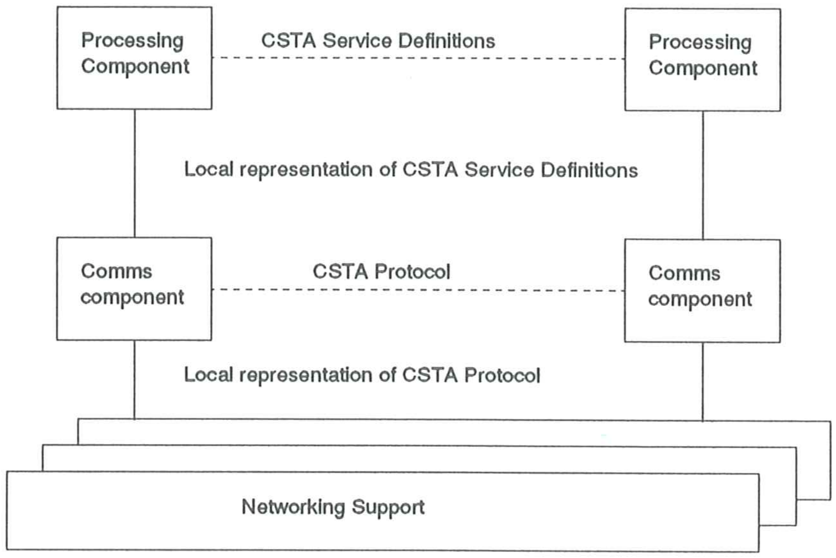


Figure 10 - Relation between Processing, Communications, and Network Support

From the figure it can be seen that the distributed application functionality interact with their peers in accordance with a CSTA service definition. In the standardization phase, these service descriptions will be formulated in a formal, abstract notation to precisely define these interactions. This formal, abstract definition can then be used to derive the service interface between the application functionality and the local CSTA service via which the peer to peer service interaction is supported. The CSTA service will communicate with its peer via a CSTA protocol (i.e. a set of messages and associated sequencing rules, etc). Each CSTA service effects this protocol via a standardized interface to local network support software.

In an OSI environment, the applications functionality and CSTA service together form an application process invocation; the necessary communications aspect would be provided by an application entity invocation considered to reside in the OSI application layer. The underlying networking support would typically be provided by OSI lower layers.

8.1.3 CSTA Service

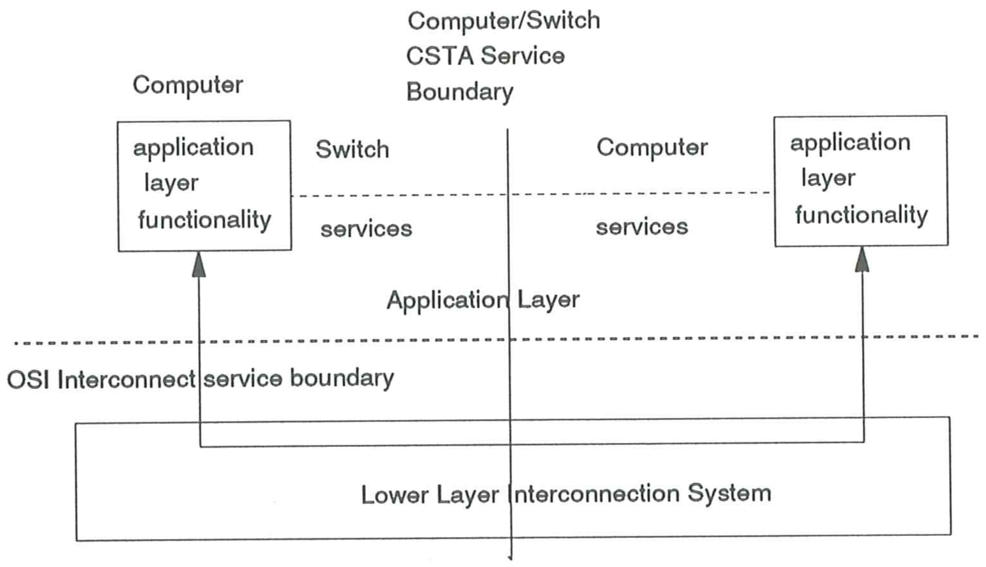
In the context of the OSI Reference Model and excluding the Application layer, the word 'service' is used to refer to the benefit provided by one layer to its adjacent higher layer.

In the context of the CCITT definition of the services provided by a real network, eg. an ISDN, the term 'service' applies to that which is offered by the network to a user at a given reference point, eg. the S reference point.

In the latter context, CCITT Rec. I.120 classifies ISDN services as either bearer services or teleservices. Each of those categories is subdivided into basic services and supplementary services. Bearer services include layer 1-3 attributes, while teleservices include those plus layer 4-7 attributes. Telecommunications services exist in the Teleservices category. Teleservice functionality can reside either within a network or within a terminal associated with the network.

The figure below shows, in simplified form, how the OSI layer and CCITT network notions of 'service' relate to one another. OSI layer services have a vertical orientation. ISDN Basic and Teleservices, as the latter also embrace those of the Application layer have a horizontal orientation.

As between these orientations, this Technical Report defers to that of CCITT and, unless otherwise qualified, uses the term 'service' to refer to the benefit provided by one application layer process to another, where the providing process may reside either fully within the switch or fully within a computer associated with the switch.



Computer/Switch
CSTA Service
Boundary
Computer
application layer functionality
Switch
services
Computer
services
application layer functionality
Application Layer
OSI Interconnect service boundary
Lower Layer Interconnection System

Figure 11 - Illustrating the OSI and CCITT Uses of the Term Service

8.1.4 Client/Server Model

The communications mechanism (as opposed to the processing) required to support the CSTA application can be modelled as a client/server relationship (such as described in ISO 10031-1). A processing component (identified in ISO 10031-1 as the User) requests a service. Its local communications component, termed a client, invokes that particular service by communication with its peer, termed a server. The client/server relationship models application level communication and hence can be considered as belonging to the OSI application layer.

Because the scope of CSTA architecture is to provide bi-directional capabilities, the client/server relationship will be possible in both directions as depicted in the following figure.

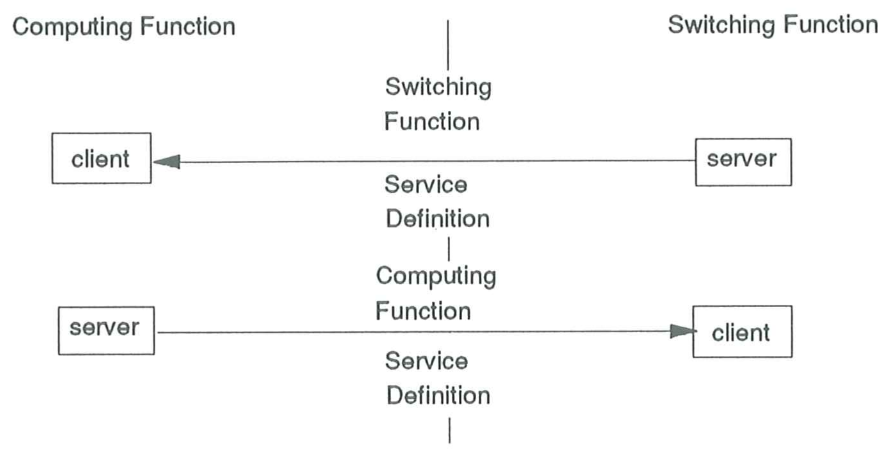


Figure 12 - Bi-Directional Service Definitions

Service definitions in which the Computing Function is the client and the Switching Function the server are defined as Switching Function Service definitions. An example of a Switching Function service is the request to establish a Call.

Service definitions in which the Switching Function is the client and the Computing Function the server are defined as Computing Function service definitions. An example of a Computing Function service is a request to convert an address.

8.1.5 Service and Objects

The service provided by a server to a client will consist of observing and acting upon objects which can be observable and accessible by the server on behalf of the client. Some examples of objects that can be involved in call requests are, the calls themselves, the terminals (devices) involved in the calls and the parties making and receiving the calls.

In order to fully define the services expected by a client from a server, there is a need to define which objects can be made visible to the client. For example, to request the establishment of a call a client needs visibility of the parties but does not need to be aware of other elements (objects) involved in the call, such as trunks.

The objects and their behaviour, as perceived over the client to server interface, will be expressed in implementation independent terms in an Operational Model (see clause 9). For example, the request to establish a call between two parties may be translated into a generic request 'Make_Call from D1 to D2'. The way the call is established might be switch dependent but any call progress information will be reported in a generic way.

8.1.6 Multi-Server/Multi-client Relationships

The definition given to the term "CSTA application" places no constraint on the way in which a client process may exploit the services of a server. Thus, a server may support multiple clients supporting one or more CSTA applications. Conversely, a single client may exploit the services of a number of servers for the purposes of supporting a single CSTA application.

8.1.7 Application Layer Relaying and Service Granularity

As described previously, a CSTA application can be distributed. In this context a number of OSI Application layer entities can cooperate to provide the end to end service. These entities, called Application layer relays, act both as servers to preceding entities and clients to succeeding entities.

In any client/server arrangement both the client and the server can, at least in principle, be indefinitely distributed in such a way that, except at the start of the chain, all entities in the chain become servers to their immediately preceding client entities.

This chained arrangement is always defined in such a way that any client does not need to know the chained (distributed) arrangement that may exist behind the server.

When the underlying communication stacks are different the application layer relay may be viewed as providing a protocol conversion.

For completeness, the concept of Application layer protocol conversion needs to be understood as embracing the concept of conversion of service granularity as observed by the different members in a client/server chain. Thus, the ultimate client may, for example, perceive the object of a service as being the end to end connection, or call, established by the server on its behalf. The server, acting in the role of client to second order servers, may see it in terms of a concatenation of lesser connections, and so on.

In the most complex arrangements a CSTA application may be viewed as a chain of application layer relays that cooperate to fulfil the application requirement as depicted in the following figure.

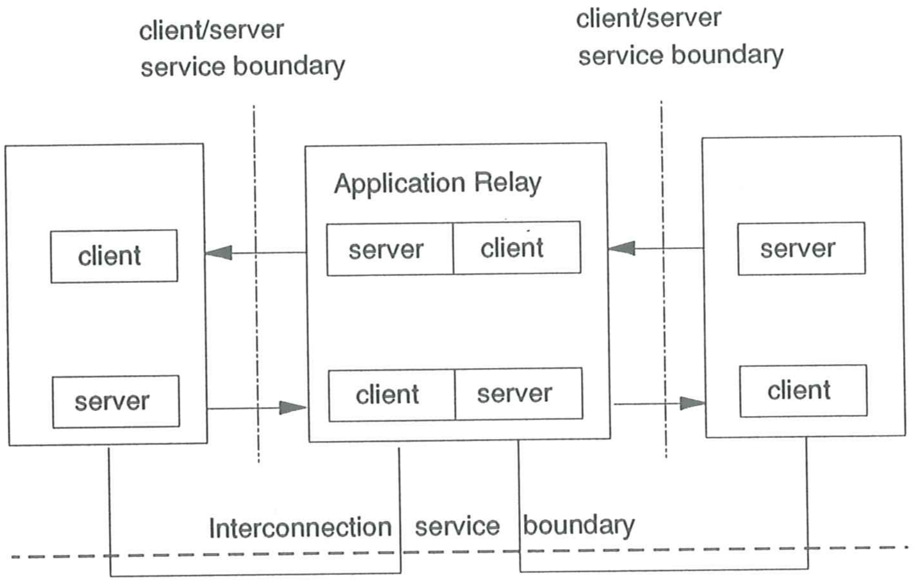


Figure 13 - Application Relay as Combination of Client/Server Functions

This figure illustrates both the concepts of Application layer relaying and the orthogonal uses of the term service.

8.1.8 Service Boundaries

The following figure depicts the service boundaries that are subject to standardization. It also shows the two domains for which an operational model is defined in clause 9.

Not shown in the following figure are the Application Programming interfaces (APIs) existing in both the Computing and Telecommunications network domains. These interfaces separate User Application Functions from the client/server functions of the Application layer existing to support them. These APIs are not regarded as being subject to standardization in this document. They are considered in more detail in clause 13. An example enlarging on the material of clause 13 is given in Appendix B.

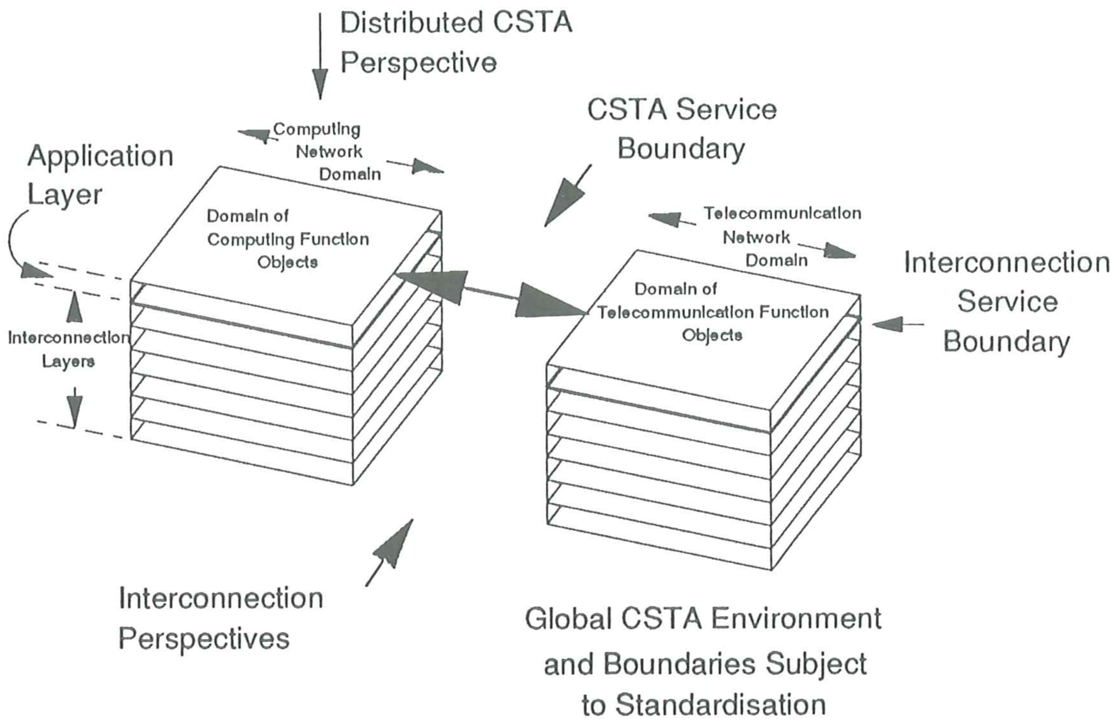

Distributed CSTA Perspective
CSTA Service Boundary
Application Layer
Computing Network Domain
Domain of Computing Function Objects
Interconnection Layers
Telecommunication Network Domain
Domain of Telecommunication Function Objects
Interconnection Service Boundary
Interconnection Perspectives
Global CSTA Environment and Boundaries Subject to Standardisation

Figure 14 - Global CSTA Environment

8.1.8.1 CSTA Service Boundary

As given previously, a focus of interest of this Technical Report is the boundary between co-operating computer and telecommunications networks, both being considered as a whole and each being regarded as a peer of the other.

This boundary is an integral part of the Operational Models (clause 9).

8.1.8.2 Interconnection Service Boundary

An inherent feature of the model, given by the orthogonal relationship between the inter-connection and CSTA Application layer architectures, is their potential independence one from the other.

Given this feature, no requirement exists in principle to maintain throughout a CSTA system a uniform interconnection architecture, OSI conformant or otherwise. It could, for example, be SS7 or, for that matter, any other proprietary interconnection architecture.

However, to meet the requirement for transportability of CSTA functions between different interconnection environments it becomes necessary to adopt, as a feature of the model, a standardized interconnection service boundary through which the inherent potential of the model can be fully realised in practice.

This boundary is regarded as a part of the Interconnection Service Architecture (clause 14).

8.2 Distributed CSTA Functionality

The functionality at the CSTA application service boundary (as shown by the large arrow of the previous figure) may be impacted by the definition and location of the Functional Entities (FEs) as a result of the application of the stage 2 methodology such as defined by CCITT Rec. I.130 and Rec. Q.65.

Stage 2 identifies the functional capabilities and the information flows needed to support a given service. A functional model is derived for each service. The functions required to provide the service are grouped into functional entities (FEs). The functional model is the aggregate of the functional entities and their relationship.

The FEs may reside in either the computer application or the switching application or both.

The mapping of the FEs onto physical locations will lead to the definition of different sets of CSTA services primitives for a given CSTA service. Clause 12 deals in more detail with this subject area.

The CSTA protocol will be designed to support the various approaches. As a consequence, some of the protocol elements will be optional and their use will be implementation dependent.

8.3 Configuration Scenarios

CSTA applications have been defined by previous clauses as distributed applications. Their reliability is ultimately determined by the reliability and topological configuration of the network components supporting them. The following considers the various configuration possibilities and their individual attributes.

8.3.1 Topologies

The topologies considered in this sub-clause should be taken as examples only and not as covering the complete set of all possible topologies.

Co-operating computing and switching networks (or their elements) may be connected in a number of ways that will not only provide different capabilities, they will also impose different constraints. In this respect all CSTA interconnection scenarios should be downwards compatible, i.e. they should be designed as true supersets of the simplest topology comprising one switch connected to one host.

Various configurations can actually be obtained as a combination of a basic set of configurations. We are primarily interested by the number of communication endpoints that reside at the edge of the computing and switching networks, not by the number of computing or switching nodes. The communication links that are used to compose the CSTA network will in fact be key elements of the topology. These will be used to implement a relationship that can be of the type one to one, one to many, many to one or many to many. It has to be noted that each link can itself be duplicated as further described in 8.3.2.1

The "one-to-one" type of scenario corresponds to the case of a point to point connection between two gateways as depicted in the following figure.

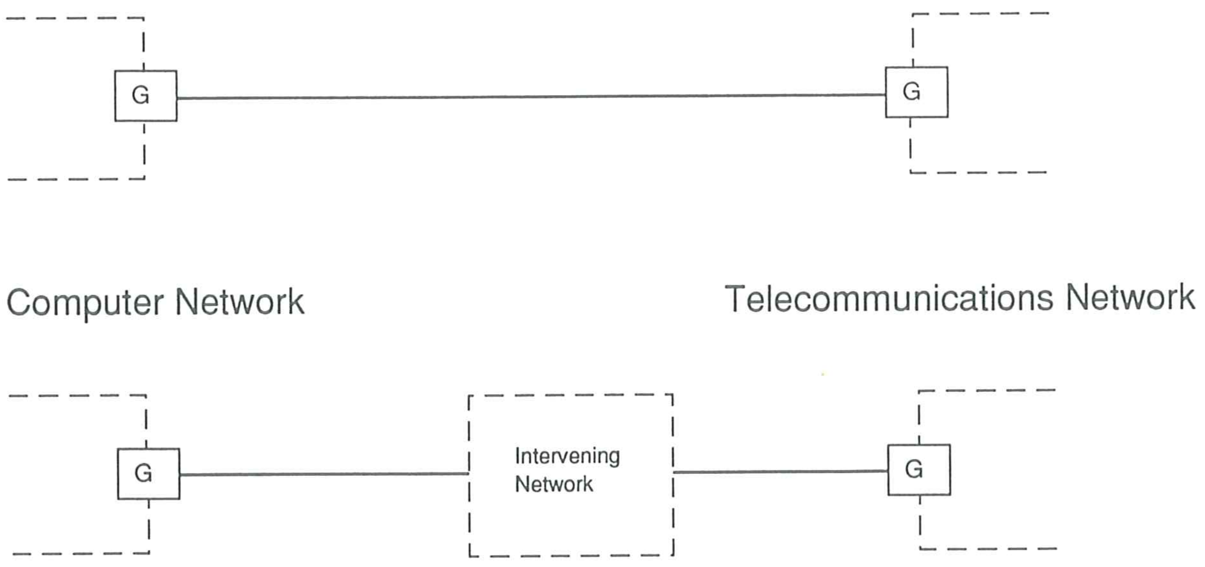


Figure 15 - Single Point to Point Configuration

The point to point connection can be implemented in various ways, e.g. via a dedicated physical link or by use of circuit or packet switched connections. This choice will affect the selection of the lower level protocols as described in clause 14.

In a "one-to-many" or "many-to-one" type of an arrangement, one gateway on one side is connected by point connections to several gateways on the other side. One server will be connected separately to several clients or one client to several servers as illustrated by the following figure.

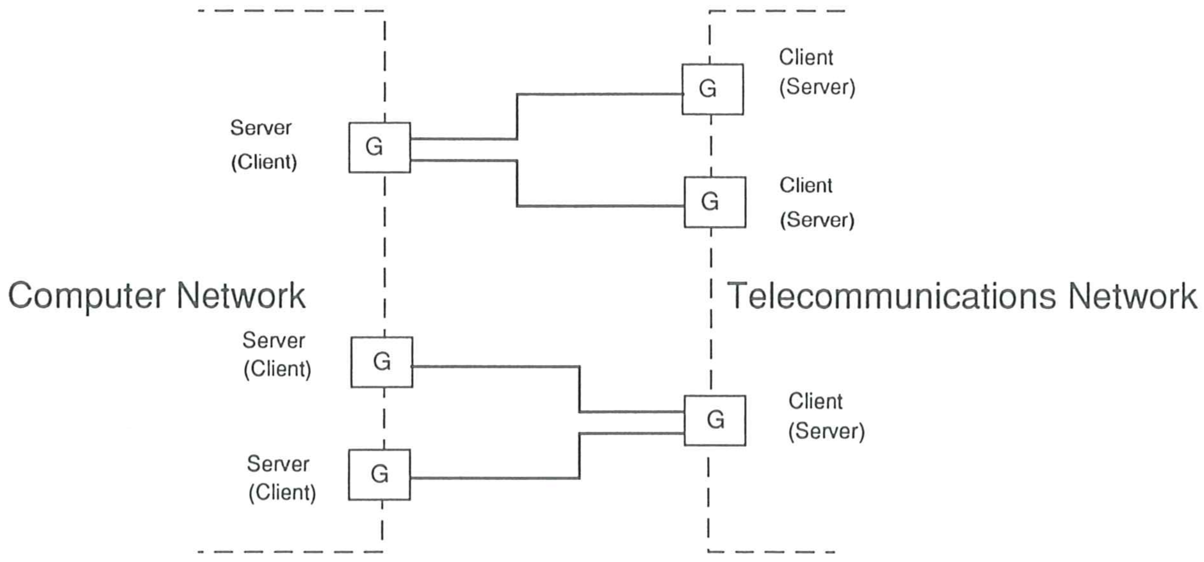


Figure 16 - One-to-Many and Many-to-One Configurations

This type of configuration does not prejudge the relative roles of the individual servers in a multi- server arrangement. These may be:

1. functionally different
2. functionally identical but logically different when they are in charge of different domains
3. functionally identical and logically identical in a redundant arrangement
4. organised to appear as one virtual server or client system (x-server, x-client in ISO 10031- 1).

In all four cases a problem needing to be solved is that of the server and client identification and addressing.

In the "many-to-many" type of scenario several gateways on one side are connected to several gateways on the other side. This can be accomplished via parallel links or via a communication subnetwork. We shall call the rest of communication facilities the communication subnetwork.

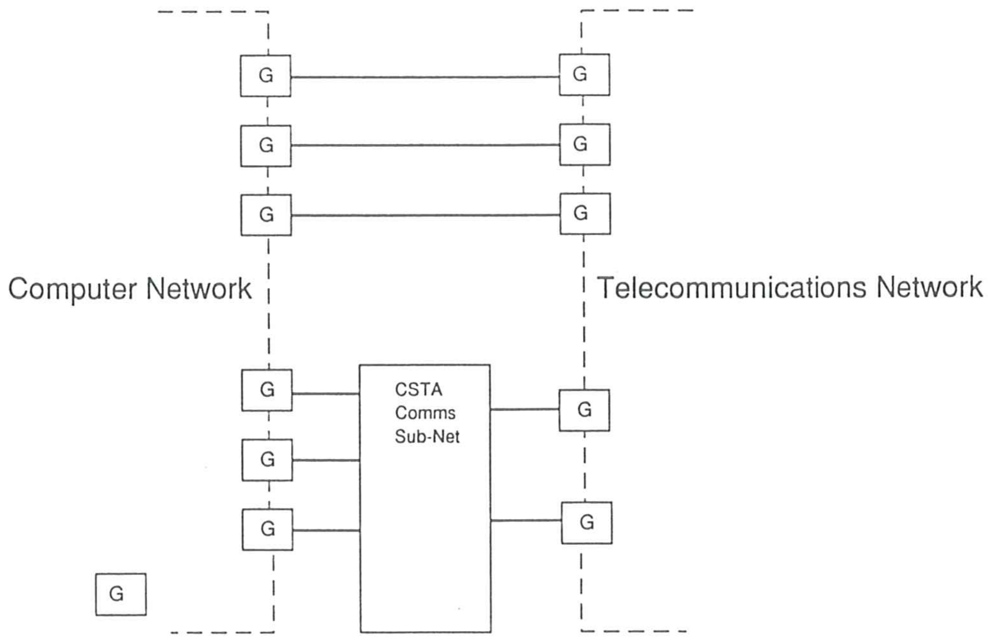


Figure 17 - Multiple Point-to-Point and CSTA Communications Sub-Network

In this latter case the communication subnetwork provides its own addressing and routing mechanisms to allow any client on one side to access any server on the other side.

8.3.2 Reliability of CSTA configurations

When computers and switches are closely interworking to provide a CSTA service to the end user, the arrangement is seen by the user as a single CSTA system. As such it is expected that this system will provide the same QoS as is normally found in either computing or telecommunications worlds. One characteristic of the telecommunications systems is high reliability.

For CSTA applications demanding high reliability, adequate solutions have to be identified. One of the standard approaches to this question is the provisioning of redundancy at various levels.

This Technical Report is mainly concerned with the communication aspects and will only briefly present what are the various aspects of, and questions raised by, the introduction of redundancy in the communication paths. These questions remain for further study. Solutions will need to be developed that allow downwards compatibility in the sense that most complicated configuration scenarios will be designed as supersets of the simpler ones.

The choice to implement one or another scenario must be part of a global strategy that depends on the actual QoS requirements for a given CSTA application. It will also integrate the reliability of each individual component. Therefore it is not the aim of this Technical Report to recommend a particular solution but rather to identify the issues and propose early directions.

The question has to be considered from two perspectives: static (meaning as a matter of physical configuration choice) and dynamic (meaning as a matter of operational status).

8.3.2.1 Static Aspects

A certain degree of redundancy can be introduced in the various configurations described in 8.3.1.

1. At the lower level, redundancy may be provided by duplicating the transmission media (most probably at link level). Multi-link procedures could adequately be used with or without a transport layer.

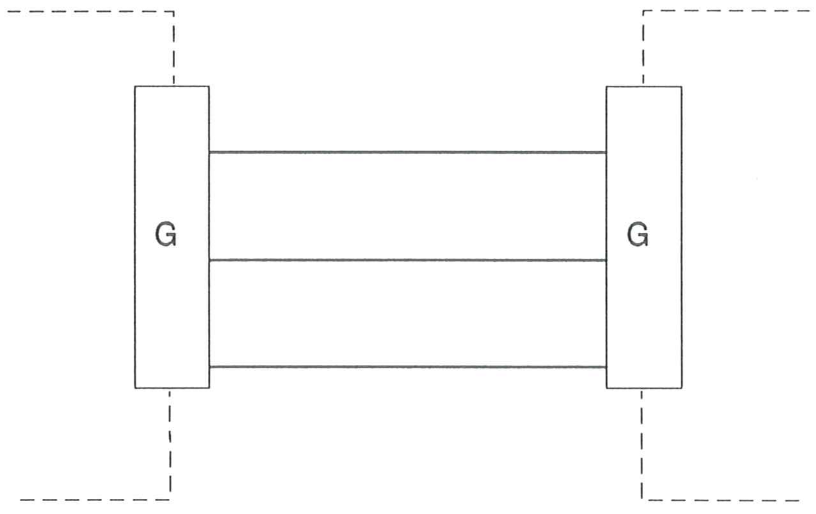


Figure 18 - Multi-Link Arrangement

2. Alternatively, redundancy can be provided at gateway level. The gateways are linked by point to point logical links that can either be multiplexed on one physical link or make use of independent physical links. One of the questions to be further discussed is the issue of accessing (i.e. naming and addressing) the gateways.

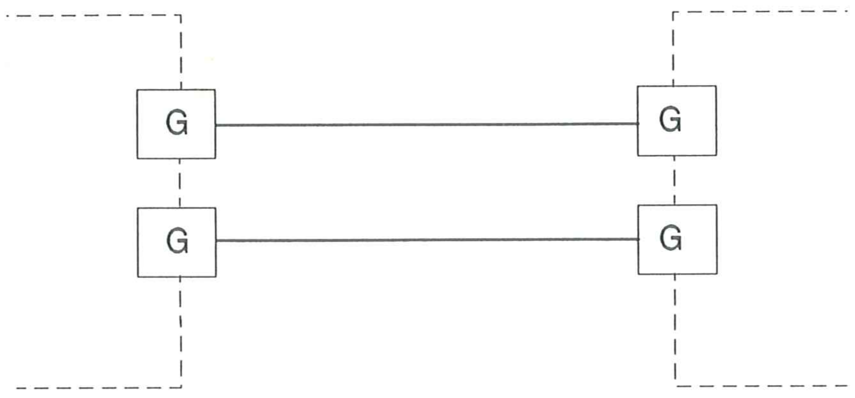


Figure 19 - Multi-Link Multi-Gateway Arrangement

3. At the communication level, further levels of reliability can be provided by installing a fully meshed communication subnetwork between the gateways. Once the naming issues, raised above, are resolved this approach is not necessarily more complicated than the previous one. It would however be more costly since it assumes the existence of an OSI network service.

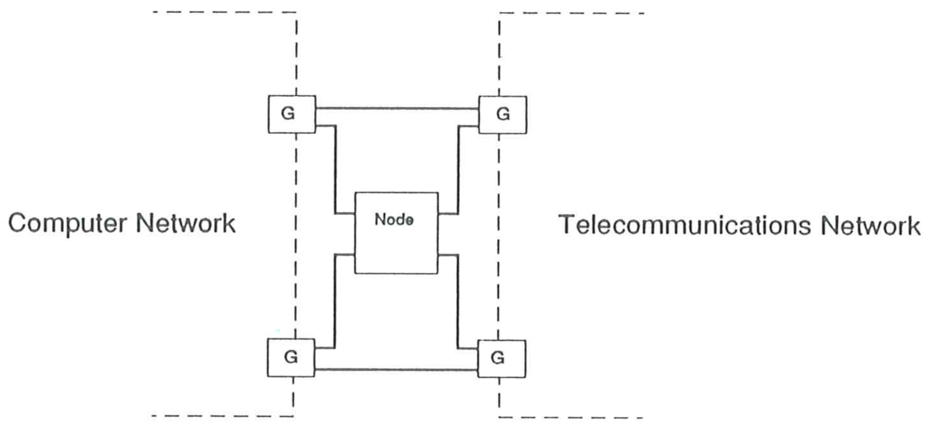


Figure 20 - Fully Meshed Communications Sub-Network

4. CSTA application processes may also be functionally duplicated, i.e. in charge of the same domain of switching objects. These will channel their commands through one or several gateways. This scenario, although more complicated than the previous ones, does not diverge from the previous ones from the communications viewpoint (and therefore standardization viewpoint). This will have to be addressed as a part of a more general question on objects belonging to overlapping domains. The interaction and potential conflicts between two Application Processes that control overlapping domains will also have to be studied.

8.3.2.2 Dynamic Aspects

In the absence of failures, all the redundant paths can be permanently active. Alternatively, only a sub-set may be defined as being active at any one time. Scenarios can range from load sharing with hot backup procedures to standby mode with change over procedures.

This topic merits some attention but is not considered as a major deliverable of this Technical Report.

It must be noted that independently of any redundancy considerations, the dynamic aspects imposed by communication failures (transient situations) will have to be encompassed by the proposed solutions, i.e. error messages and recovery procedures will have to be defined that will allow an application to cater for the loss of information messages. This is an issue that pertains to communication management, as such it will be discussed in the relevant clause.

As a general principle, solutions will have to be developed that will allow backwards and downwards compatibility with the simple configuration which is the prime object of this Technical Report.

9. OPERATIONAL MODELS

9.1 CSTA Domains and Sub-domains

Due to the limitations of existing signalling systems, a computer or a switching network may not present the appearance and functionality of a single computer or switch in terms of the way in which it responds to CSTA service requests.

Consider the situation of an application on a single computer which monitors two devices on two switches. The computer is not connected to any other computers. The two switches are connected to each other by a signalling system such that they do not present the appearance and functionality of a single switch to the computer. They are not connected to other switches. The application process in the computer is monitoring the devices D1 and D2. A CSTA Make__Call request is issued such that a call is initiated from D1 to D2.

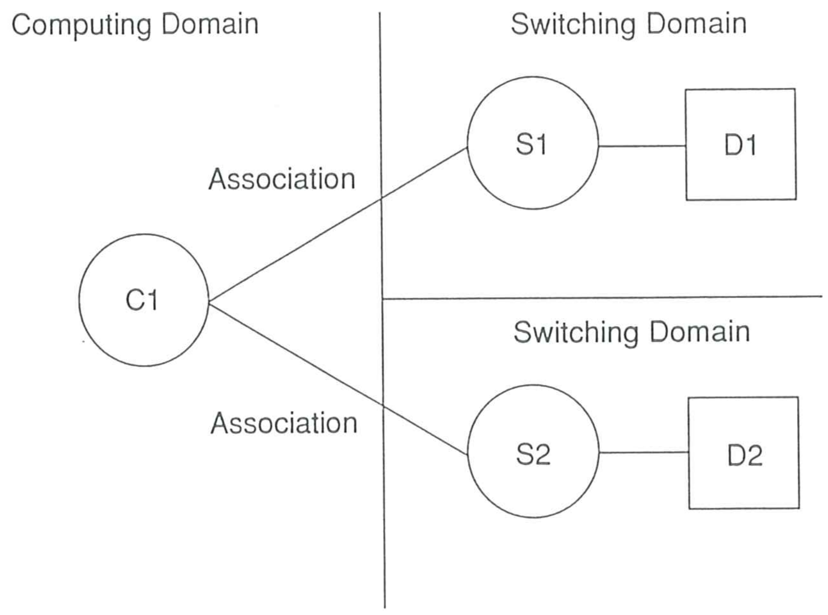


Figure 21 - Consideration of Computing and Switching Domains

Due to the nature of the signalling systems between the switches, S1 and S2, it will be known at S1 that D1 is connected to a trunk, similarly it will be known at S2 that D2 is connected to a trunk. It will not be known that D1 and D2 are connected.

It should be noted that similar situations can arise in the computing domain.

The introduction of additional terminology provides a framework to express the problems which arise as a result of the limitations of some signalling systems.

9.1.1 Definitions

A switching domain is the set of all the switches and their objects which may be reached directly or indirectly by a CSTA application from a computing domain.

A computing domain is the set of computers and their objects which may be reached directly or indirectly by a CSTA application from a switching domain.

A switching sub-domain is any configuration of inter-connected switches which presents the external appearance and functionality of a single switch to the computing domain.

A computing sub-domain is any configuration of inter-connected computers which presents the external appearance and functionality of a single computer to the switching domain.

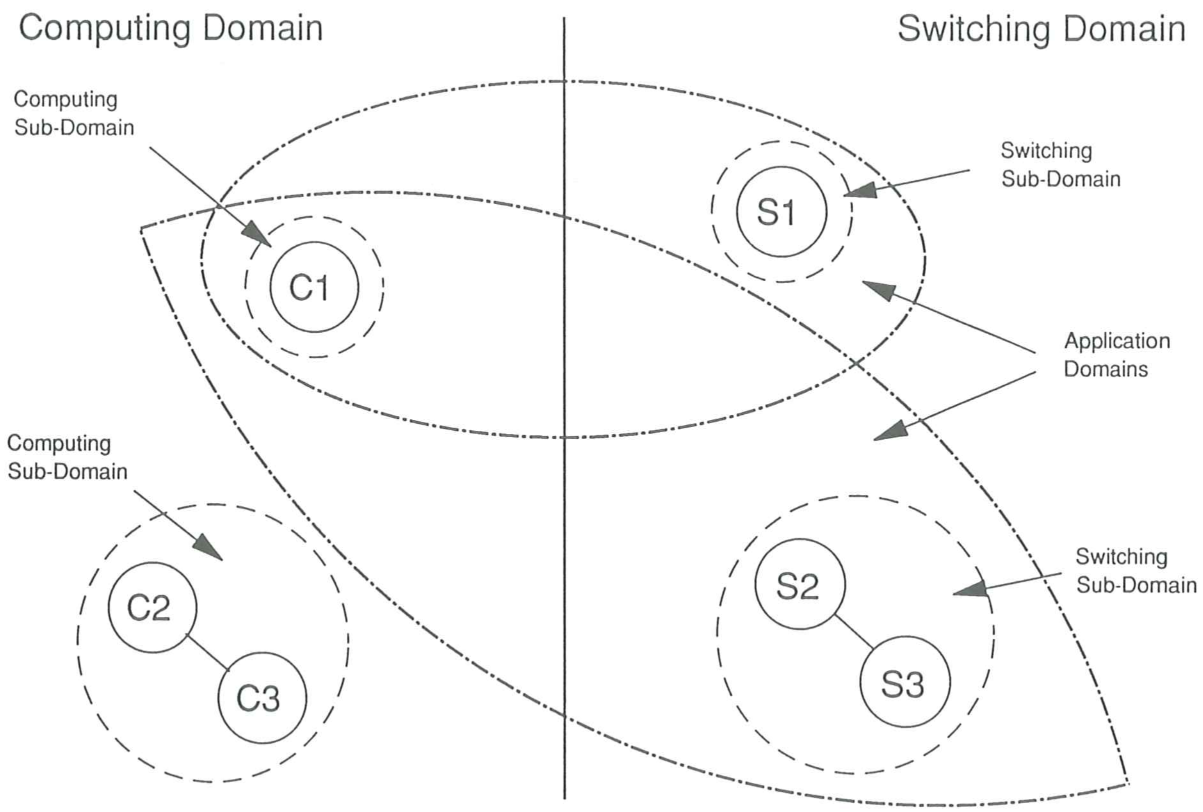


Figure 22 - Domains and Sub-Domains

An application domain is the union of one switching sub-domain and one computing sub-domain. A sub-domain can exist in any number of application domains.

The set of application domains for an application is called the application environment. An application domain may exist within more than one application environment. The mechanism for developing a consistent view within an application environment is not within the scope of CSTA. In the example given at the start of this clause it is beyond the scope of CSTA, for example, to deduce that D1 is connected to D2, since although D1 and D2 are in the same Application Environment they are in different Applications Domains.

A computing network corresponds to either a computing domain or a computing sub-domain. Similarly a switching network may correspond to either a switching domain or a switching sub-domain. If the signalling within a network is such that it represents the appearance and functionality of a single switch or of a single computer, that network is a sub-domain.

A Computing Function corresponds to a computing sub-domain.

A Switching Function corresponds to a switching sub-domain.

The model introduced in Clause 8 (Functional Architecture) defines the CSTA objects and their behaviour as perceived at the client/server interface.

In the example given at the start of this clause, the union of S1 and S2 is the switching domain. S1 and S2 are said to be in different sub-domains. The single computer represents both the computing domain and the computing sub-domain. The union of the computing sub-domains and the switching sub-domains form the CSTA application environment.

9.2 Switching sub-domain Model

CSTA defines the ability for an application to act on behalf of one or more users. To accomplish this for users of a telecommunications network it is recognised that some applications will need to monitor telecommunications activity to operate intelligently. Consequently, many possible indications given to users of a telecommunications network may also be of interest to CSTA applications.

The Switching sub-domain Model defines the tools needed to provide an abstract view of the telecommunications network. This model allows application conceptualisation of the telecommunications network's operation. To provide this abstract view CSTA defines several CSTA Switching sub-domain Model Objects that may be observed and acted upon by the Switching Function on behalf of the Computing Function.

9.2.1 CSTA Switching Model Objects

9.2.1.1 CSTA Device

CSTA will not be able to influence and provide unambiguous information about parties participating in calls. It will, however, be able to influence and provide information about the access points of the telecommunications network. Access points in general are known as CSTA Devices, and these telecommunications access points are known as CSTA telecommunications Devices.

Definition: A Device is a logical entity which interfaces the actions of entities external to the telecommunications network (typically a Party or User) to a telecommunications service providing application (e.g. telecommunications sub- domain).

The CSTA telecommunications Device is used to refer to both physical and logical devices (such as buttons, keypads, lines, trunks, telephones, etc.) and logical entities (such as extensions, hunt groups, ACD groups, etc.). Examples of various types of telecommunications devices are given below.

9.2.1.1.1 Device Attributes

1. Device Type - a number of different types of device have been identified as being important to CSTA:

   - Public Directory Number - a number assigned to various physical devices to allow them to become addressable by the public network. An E.164 number.

   - Private Directory Number - a number (extension) assigned to various physical devices within a private network.

- Button/Line/Station - it is desirable to be able to identify an individual station. In some situations it may be desirable to identify a given button or line on a multi-line station or an individual station where several stations share a single Directory Number. The method for identifying these is for further study, but may involve the use of sub-addressing.
- Trunk/Trunk Group - in order to manipulate and view calls that cross a CSTA switching sub-domain it may be desirable to address the point at which the call crosses the boundary. This point will generally be a trunk or trunk group. The method of identifying trunks and trunk groups is for further study.
- ACD/Hunt Group - in order to interact with mechanisms that distribute calls within a telecommunications network, it may be desirable to address the group as a whole, or the distribution mechanism as a part. These groups are often assigned Directory Numbers (Pilot Numbers). The static identification of ACD/Hunt groups with no single Directory Number is for further study.

2. Device Identifier - each of the different device types will need to be referenced across the CSTA Service boundary. To accomplish this, each will require a device identifier. Two types of device identifiers may be required:

   a. Static Device Identifier

      In some cases the identifier is stable over time. This identifier remains constant and unique between calls, associations and perhaps even CSTA application domains. This Static Device Identifier is a form of identifier which is known a-priori by both the Computing and Switching Functions. An example of a Static Device Identifier is a Directory Number.

      It may also be useful for the Switching Function to convert this identifier to another static form for subsequent use in service interaction. An example of this would be the transformation of a Public Directory Number to a Private Directory Number. This transformation allows:

      - service interactions to be independent of the identification mechanism
      - optimisation in the amount of data exchanged.

      A Static Device Identifier is normally applicable throughout the CSTA application domain. Having received a Static Device Identifier, the Switching Function may have to undertake a number of steps before an actual physical device is selected. As an extreme example, if the Static Identifier is a hunt group pilot number, then the Switching Function will start by resolving it to a particular member of the hunt group. The Switching Function may then resolve diversion assigned to that member of that hunt group. Finally, if the terminal endpoint is a multi-line set, the Switching Function may have to select a particular line before the actual physical device is determined.

   b. Dynamic Device Identifier

      Once a device has been included (resolved) in a call, it may be desirable to continue to refer to it for the purpose of manipulation or tracking. The Static Device Identifier may not always be sufficient for this purpose because:

- for various reasons a Static Device Identifier is not available. (There is no a-priori identifier for the particular device)
- the Static Device Identifier is too long and cumbersome for efficient use.

In these cases the Switching Function assigns a Dynamic Device Identifier to be used as a handle for the duration of the call. Management of the Dynamic Device Identifier is discussed later in this clause.

3. Device State - the set of connection states which are associated directly with a particular device. For information about connection states see the sub-clause on Connections later in this clause. For more information about Device States, see Clause 11 - States and Events.

9.2.1.2 CSTA Call

Definition: A Call is a Switching Function communications relationship (generally) between two or more parties. During some circumstances, including set-up and release, there may be only one party.

Calls, including their establishment and release, may be observed and manipulated across the CSTA boundary. During some phases of the call (e.g. establishment and release) the call is not completely formed and there is often a single device involved (for example, the device that requested the call). In many operations, such as transfer, one device in a call is replaced with another device. In these situations, a CSTA call is maintained as long as the telecommunications relationship remains across each operation.

9.2.1.2.1 Call Attributes

1. Call Identifier - a Call Identifier is allocated to each call by the Switching Function when it first becomes visible across the CSTA Service Boundary. It may or may not be globally unique within a switching sub-domain. To allow reference to a nascent call, the call identifier may be assigned before the call is established. For example, an incoming call may be assigned a call identifier when the called device is alerting and before the call has been answered. This call identifier references the entire call within the sub-domain.

   The CSTA call may pass through various stages involving many and different devices before it finally terminates. Examples of CSTA services that cause this evolution are Transfer and Conference. During these operations the call identifier may change. The management of the call identifier is described in a following sub-clause.

   The Call Identifier is made visible across the CSTA service boundary in (at least) three ways:

   - it is returned by the Switching Function on acceptance of CSTA requests that initiate calls (such as Make__Call)
   - it is provided in Monitor service event reports about a call (a new incoming call would be identified this way)
   - it can be obtained by using the Snapshot service on a Device.

2. Call State - the set of Connection states for those connections which comprise a call. For more information on Connection states, see the following sub-clause on Connections. Call States are described in more detail in Clause 11 - States and Events.

This set of connection states will be encoded in a form which is easily interpretable as a call state.

9.2.1.3 CSTA Connection

Definition: a Switching Function relation between a device and a call.

This relationship is both observed and manipulated. In fact, observation and manipulation of these relationships make up many CSTA services (e.g. hold, reconnect, drop). Connections are CSTA Objects.

9.2.1.3.1 Connection Attributes

1. The connection identifier is defined as a tuple of the Call Identifier and Device Identifier. It is unique within a sub-domain. For a single association there will be a unique Connection Identifier. Different associations may have different Connection Identifiers for the same connection.

   For a call there are as many connection identifiers as there are associated devices, and for a device there are as many connection identifiers as there are associated calls.

2. Connection State - one of a set of states a connection may have. Connection states are reported by Snapshots on either calls or devices, and changes in connection states are reported as events by Monitors. The connection state refers to a single Call/Device relationship. A simplified connection state model is given below.

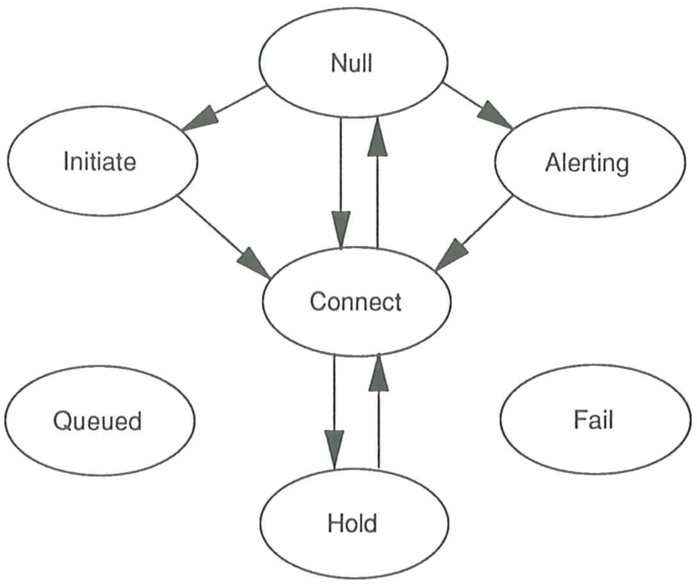


Figure 23 - Connection State Model

In this figure, the states presented are envisioned as a basic set. The transitions between states, shown by arrows, show the states possible to enter from a given state and form the basis for providing events when they occur. These states are not equivalent to ISDN access states. They are a derivation of the state machine on one side of an ISDN access. A brief description of these states follows, but for more information about states and events, see Clause 11 on States and Events.

Null - the state where there is no relationship between the call and device.

Initiate - the state where the device is requesting service. Usually this will result in the creation of a call. Often this is the "dialling" state.

Alerting - the state where a device is alerting (ringing). This usually indicates that a call wishes to Connect to a device.

Connect - the state where a device actively participates on a call. This state includes the notion of a logical participation with a call as well as a physical participation to that call.

Hold - the state where a device inactively participates on a call. This state embodies the notion of a logical participation with a call with a suspended physical participation to that call.

Queued - the state where normal state progression has been stalled. This state generally refers to two conditions but can apply to others as well. One condition is when a device is trying to establish a connection with a call, and the process is stalled. The second condition is when a call tries to establish a connection with a device and that process is stalled.

Fail - the state where normal state progression has been aborted. This state generally refers to the condition when a device tries to establish a connection to a call and the attempt fails. This can fail because of many reasons including: failure on connection between calling device and call; failure on connection between called device and call; failure to create call.

Note once again that these states are preliminary. These states apply to one device on one call, and not to any other devices that may be involved in that call. Also, states like Fail and Queued are for further study.

9.2.1.4 Relationship between Switching Sub-domain Objects

An overall model of a switching sub-domain is composed of a set of relationships between all calls and devices within that sub-domain. If this set is organised so that calls appear as rows of a matrix, and devices appear as columns, all relationships between calls and devices for a telecommunications sub-domain are described. For example:

	Calls							
Devices	C1	C2	C3	C4	C5	..	..	Cm
D1								
D2								
D3								
D4								
D5				A				
..								
Dn								

Figure 24 - Device/Call Matrix

The matrix above describes a telecommunications network comprised of n devices (D1-Dn) and m possible calls (C1-Cm). Device D5 is related to call C4 in the way described by their connection (D5:C4). In this example the state of the connection (D5:C4) is Alerting .

Using this model, devices can be described by the number and type of relationships that can be supported. The simplest devices in a telecommunications network can have only one non-null state. That is, they can be related to no more than one call. Other devices can be related to more than one call, but have restrictions in the state of those relationships. Some devices can actively participate in many calls in any connection state.

Most telecommunications networks have similar, but less well known limitations for calls. For example, basic calls relate exactly two associated devices, while more complex calls may relate a larger but generally fixed number of devices.

It may be sufficient for applications in the computing domain to identify a connection by specifying only the call or device identifier (static or dynamic) part of the connection id provided that the information provided satisfies the requirement that the connection be uniquely identifiable within the switching sub-domain. In these cases it is possible for the switching sub-domain to infer the proper paired identifier from the context of the request. For example, for a telephone that can only participate in a single call, it may be possible to refer to the connection between that telephone and call using only the device identifier. Such a telephone which could only participate in a single call might be represented by the following matrix:

	CALL C1
Device D1	Any State

Figure 25 - Typical Matrix Representation

For device D1, call C1 is the only call for which a connection relationship is present. Any operations for this connection can be inferred by giving the device identifier and operating on the only associated call.

Similarly the switching sub-domain is free to construct the Connection_id in any form it chooses, provided that it satisfies the requirement that the connection be uniquely identifiable in the switching sub-domain. Therefore the switching sub-domain could construct Connection_ids containing only a call identifier part or a device identifier part given that it satisfies this uniqueness requirement.

9.2.1.5 Derivation of Call Events and States

The purpose for the switching model just presented is to provide an abstract of actual states and events that are communicated via underlying signalling systems. This abstract view is probably more detailed than required by CSTA applications, but is presented to introduce a more exact language for describing CSTA events, states and service actions.

Because of the topology of the telecommunications network, the signals that report events and changes in states have definite sources. Providing a telecommunications object (the connection) that can be associated with the source of these signals helps in describing the meaning of the events and the operation of CSTA (and other) telecommunications services.

On a typical ISDN access to a network there exists a distributed state machine. One part of this access state machine resides in the ISDN device. Another part resides on the other side of the ISDN access. There is another similar distributed access state machine which resides across the ISDN network at a similar device. One of these access state machines was abstracted and just presented.

Using this concept, a call can be modelled as a collection of connection state machines communicating with one another using signalling. When this communication occurs, a CSTA event can be generated. In the following example, this concept of communication between two state machines is demonstrated for the case of establishing a simple call. Additionally, on either side of the example the ISDN call states have been provided.

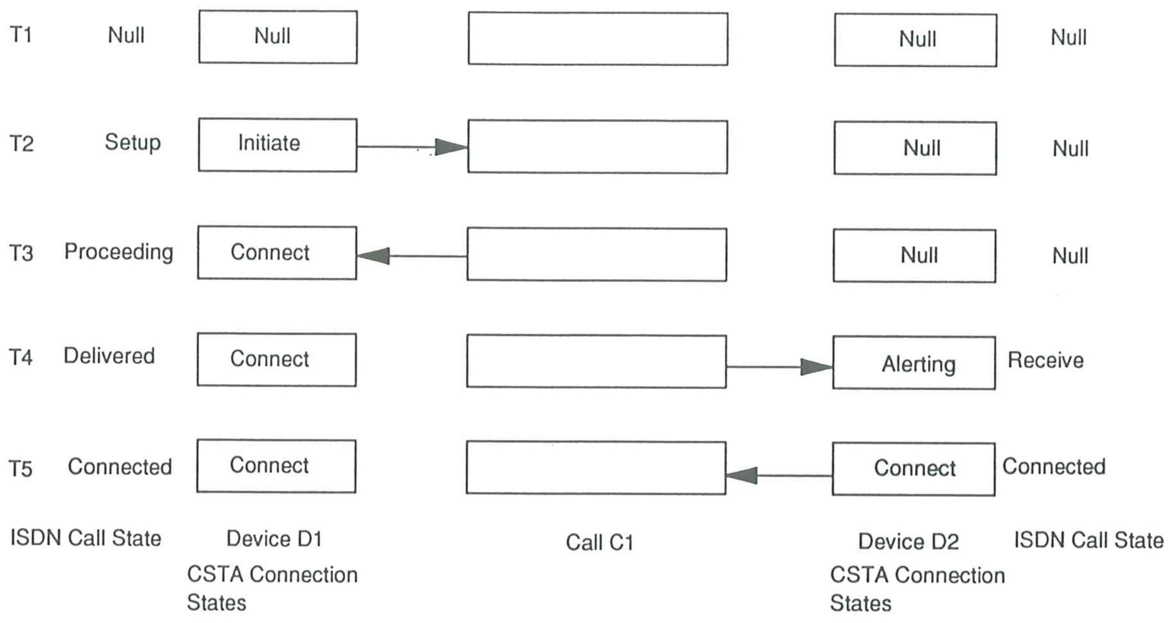


Figure 26 - Call Progression Example

Each of the states in this progression can be shown in the compact form demonstrated in the matrix shown here:

Call C1 at time:

		t1	t2	t3	t4	t5
Devices	D1	N	I	C	C	C
	D2	N	N	N	A	C

Figure 27 - Call State Progression

In this matrix the rows show devices and columns show the progression of call C1 over time. From this a simple table can be constructed that relates the pairwise states of two devices to the ISDN call states.

Connection	State	ISDN Call State
N	N	Null
I	N	Setup (Initiate)
C	N	Proceeding
C	A	Delivered (or Received)
C	C	Connected

Figure 28 - Relation between Connection States and ISDN Call States

Many connection events are of interest to CSTA applications. Typically, however, a CSTA application is interested in atomic telecommunications activities, and often these involve many connection events. Generally, telecommunications operations embody changes to many connections. These events can be summarised in a single call event. For instance, the telecommunications services transfer, conference and clear call all perform changes in multiple connection state machines. To make this relationship clearer, the following examples are provided:

This example shows how the event Call__Cleared relates to multiple connection events for multiple devices on a single call.

Call C1 at times:

Devices		t1	t2
	D1	C	N
	D2	C	N
	D3	C	N

Figure 29 - Clear__Call CSTA Event

This example shows how the event Call__Conferenced relates to multiple connection events for multiple devices on multiple calls. Because this event affects 2 calls, two matrices have been provided to show before and after times.

Calls at time t1

Devices		C1	C2	C3
	D1	H	C	N
	D2	C	N	N
	D3	N	C	N

Calls at time t2

Devices		C1	C2	C3
	D1	N	N	C
	D2	N	N	C
	D3	N	N	C

Figure 30 - Call__Conferenced CSTA Event

In clause 11, a set of CSTA events will be presented and defined using the connection matrix format. These events are presented to show the types of interactions of interest to CSTA applications and are not intended to constrain the definition of the Monitor service in clause 10.

9.3 Computing sub-domain Model

A mature computing sub-domain model will be provided in Edition 2. The following material is a collection of preliminary information from discussions that have taken place up to now.

9.3.1 Introduction

The computing model is envisioned to support CSTA services that are not currently defined or subject of standardization in other areas. For example, it is not the purpose of the computing model to re-standardise existing methods for remote file access, database manipulation and message handling. It is envisioned that there are a set of computing services that are specific to CSTA (like Routing and Data I/O) which require a reference model to support. This computing model is intended to provide the reference model for support of those services.

The computing model describes the abstract CSTA Objects that may be observed and acted upon by the Computing Function on behalf of the Switching Function. The definition and description of these objects and their general behaviour and attributes are now described.

One aspect of the computing model is the requirement to include within it the equivalent of the switching model device for the purposes of supporting the integrated Switching/Computing Function interface to the CSTA user interface as illustrated in Fig 1.

9.3.2 CSTA Computing Model Objects

Criteria for selecting CSTA Computing Objects:

CSTA objects forming part of the computing model must be either visible or capable of being influenced across the CSTA service boundary.

Definition: Switch-party: An entity outside the Computing Function which has the intelligence to use the Computing Function.

The CSTA computing objects include the following:

- Software Applications. These are software programs residing on top of the Computing Function. Every application has a unique global identifier.
- File Directories. Every file directory has a unique identifier.
- Files. Every file has an identifier that is unique within a file directory.
- File Systems
- Databases
- Terminal/End Points/Computer Ports

9.3.2.1 Computing Device

CSTA will be able to provide information about the points of access of the Computing Function. These points of access are modelled as CSTA computing Devices. Devices can be either physical or logical. Examples of devices include:

- Terminals
- Printers
- Ports
- Systems

9.3.2.1.1 Device Attributes

Device Identifier - the device identifiers for computing devices are envisioned to be similar to those of the telecommunications network. Static Device Identifiers are long term identifiers that refer uniquely to a device, and Dynamic Device Identifiers are "handles" provided by the computing network as a short form identifier for a device. As with the telecommunications network, Dynamic Device Identifiers are valid only within the context of the association in which they are provided.

NOTE 3
Dynamic device identifier will be investigated during the later development of the computing model.

9.3.2.2 Data

CSTA will be able to provide as well as manipulate data as a Computing Function. Data can be modelled as an attribute of a device, (for example, data could be the attribute of a file) but in the general case, data can be distributed across many devices and still represent a single observable, manipulable object. In this case data is a CSTA object.

Examples of Data Objects are:

- Database
- Record
- Field
- Application Instructions

CSTA will allow service requests from the switching domain to read and write data in the computing domain. This will require the switching domain to have a model of the data within the computing domain.

An object oriented view of data is one possible approach.

The switching domain views data within the computing domain as objects and sets of objects. An object orientated view of data gives a high level of independence from the database structures and the type and configuration of computers used.

An object possesses data and the operations which may be performed upon it. For example, a list of names which a user may read and write is an object whose data is the name list and whose operations are read and write.

9.3.2.2.1 Data Attributes

Data Identifiers - data can be identified with the same scheme of Static and Dynamic identifiers that is used for Devices.

9.4 Dynamic Identifier Management

Management of Dynamic Device Identifiers and Call identifiers is provided in the following manner. Management of connection IDs is provided through management of dynamic device identifiers and call identifiers. The management scheme applies to both the switching and computing models.

Identifiers are provided when they are created. When a call is made its identifier is provided. It is also provided in event reports that pertain to the call. When a device becomes involved in a call the dynamic device identifier is provided in the events that occur at that device.

Identifiers are updated when needed. If a call changes its identifier when a conference or transfer occurs, an indicator is provided that links the old call identifier to the new identifier. Similarly, if a dynamic device identifier is changed, the new identifier is provided and linked to the old identifier.

Management of identifiers is for further study.

Identifiers cease to be valid when their context vanishes. If a call ends, its call identifier is no longer valid to refer to that call. Similarly, if a device is removed from service or from a call, its dynamic device identifier may become invalid.

Identifiers can be reused. Once an identifier has lost its context it may be re-used to identify some other call, device or data.

Call and Device Identifiers are not guaranteed to be globally unique. CSTA requires that the combination of Call and Device Identifier be globally unique within a CSTA switching sub-domain. To accomplish this, either the Call Identifier, or the Device Identifier (or both) should be globally unique.

10. CSTA SERVICE DESCRIPTIONS

10.1 Introduction

This clause provides the Stage 1 definition of CSTA services. It is concerned with the interactions at the CSTA Service Boundary of CSTA messages that flow between the computing sub-domain and the switching sub-domain. The services are defined in terms of "what" the service accomplishes, not "how" the service is provided.

The services are modelled, where appropriate, on the CCITT stage 1 descriptions of ISDN Supplementary Services contained in the I series Recommendations.

10.1.1 General Procedures

10.1.1.1 Requirements for Service Descriptions

The following requirements are distilled from the Application Examples clause. The set of switching sub-domain and computing sub-domain services provided within the overall CSTA architecture, should be extensive enough to satisfy these requirements. There is no implication in this text that all of these services and service parameters are available in conjunction with all applications either on the switch or the computer. The requirements are:

1. The operations supported, and the way in which they work, should be independent of any particular switch and reflect the defined operational model.
2. Each operation should conform to a service description that defines:
   - the valid conditions for use of the service;
   - the possible results of invoking the service (including what events may be reported);
   - the possible exception conditions (e.g. what can go wrong);
   - the interaction with other CSTA operations;
   - the example interaction with other network conditions;
   - The service definition should not depend on the switch to computer, signalling nor a particular voice terminal communications signalling protocol.
3. It should be possible to issue requests on behalf of an extension (e.g. initiate a call from a specified voice terminal, invoke Do Not Disturb for a particular extension).
4. It should be possible to issue requests that operate on a particular call (e.g. retrieve a specified held call, clear one (of several) calls associated with a particular device).
5. It should be possible to execute concurrent requests for different objects. The results of executing concurrent requests for a particular object are undefined.

6. It should be possible for an application to exploit a particular feature unique to the individual switch so that switch independence becomes the responsibility of the application (an example of such a feature is where a preset conference can be established via a single request).

7. Requested functions should operate the same way irrespective of whether Event Reporting has been selected for the associated extension.

8. It should be possible to designate calls in priority mode. Examples of this are:

   - the ability to differentiate between priority calls and routine calls, because emergency calls may require different outgoing call algorithms;
   - the ability to select a level of precedence for the request (for example, to cater for busy trunks);
   - the ability to force delivery of the call to the destination (for example, when encountering busy or "Do Not Disturb" conditions).

10.1.1.2 Operation of the CSTA Protocol

If a service is available within a particular application domain, then a service request can be issued by the client. The server verifies that a service request is valid and correct, and notifies the client in order to acknowledge or reject the service request. Verification of correctness means that the request is syntactically correct (i.e. the parts are consistent with one another and consistent with other system information, like directory numbers, and with the states of the CSTA objects involved). For the situation where the server cannot complete the service action once having acknowledged the service request, the protocol must support an event report that will indicate that the service cannot be completed.

10.1.1.3 Correlation of Responses and Events with Requests

The CSTA service requires two distinct types of correlation:

1. Relating a single response to the original request (for example, in all the call-related services).

2. Relating multiple reports to the original request (for example, in the Monitor service).

This correlation support can be provided by the underlying protocol infrastructure (for example, the ROSE Invoke ID and the ROSE child operations). The adoption of this underlying support or the provision of other mechanisms, is for further study during the standardization phase.

10.1.1.4 Control and Management of CSTA Services

CSTA identifies two methods for the control and management (i.e. activate, deactivate or reconfigure) of CSTA services. The first method is that the control of the service is provided by the application process on the server (e.g. a control and management capability is within the server domain). The provision of the control capability in such a context is known to the client by pre-arrangement. The second is that the control of a service is by an explicit service request from the application process on the client. Both methods can be used for CSTA applications.

For the case of reporting CSTA events between the Switching Function and the Computing Function, these two methods can be described as follows:

1. Event reports may be provided according to an established application context, such that the control of event reports is provided by the server application process (AP). The AP receiving the event reports does not request the service; the request

is implicit. For example, in the case of the monitor service, the switching AP could send event reports over the CSTA interface without the computing AP invoking the Monitor request.

2. Event reports may be requested dynamically, such that when event reporting is required, the AP wishing to receive event reports explicitly requests the service from the AP providing the event reports. For example, the computing AP could use the Monitor Request to request that the switching AP report events.

   Note that according to the client/server model, the AP providing the event report would be the server and the AP receiving an event report would be the client.

10.1.1.5 Opinions Regarding Server Method of Operation

There are divergent opinions on how a service should be provided. One opinion is that all servers have to provide a CSTA service in exactly the same way. Another opinion is that individual servers may react differently and inform the client what was actually done. The general approach of this clause is to provide parameters for such options so that the user of CSTA can choose the desired method of operation, and it is up to the service provider to determine whether it can provide the explicit service. It is for further study whether parameterisation of options is adequate for a particular service or whether the distinction between those options can only be accomplished by distinct services. The Technical Report currently assumes that parameterisation is adequate.

The determination of mandatory versus optional information elements will be determined in the next stage of standards definition.

10.1.2 Categories of CSTA Services

The CSTA services are divided into the following categories:

1. Switching Function Services are those provided by a switching network application, implemented in one or more telecommunication switches.
2. Input/Output Services - this category contains services that involve using a Switching Function terminal as a data input/output device.
3. Computing Function Services are those provided by the Computing Function and invoked by the Switching Function.
4. Bi-directional Services can be provided by either the Switching Function or by the Computing Function.
5. Status Reporting Services - these relate to overseeing the progress of a call-related service.

10.1.3 Graphical Representation

This sub-clause describes the graphical conventions used in the service descriptions.

A FULL STOP character stands for a CSTA device.

A capital letter "D" followed by a number is the CSTA device identifier.

A line of any sort represents a connection; the state of the connection is indicated by the form of the line:

1. A dotted line represents the connection in the initiating state.
2. A solid line represents a connection in the connect state.
3. A dashed line represents a connection in the hold state.

4. An arrow represents a connection in the alerting state.

5. The absence of a line represents a connection in the Null state.

An ASTERISK character represents a CSTA call.

A capital letter "C" followed by a number is the call identifier.

A two-device call configuration between devices D1 and D2 is shown in the following figure.

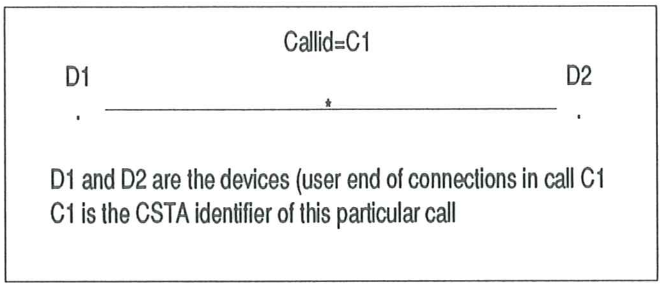

Callid=C1
D1
D2
D1 and D2 are the devices (user end of connections in call C1
C1 is the CSTA identifier of this particular call

Figure 31 - An Active Call between Devices D1 and D2

The hold relationship of a call to a device is shown in following figure as a dashed line.

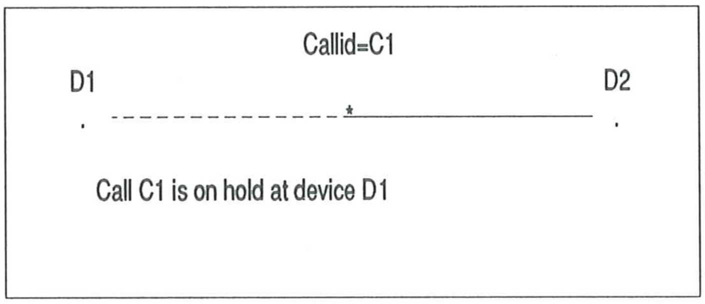


Figure 32 - A Call Held at Device D1

10.1.4 CSTA Application Private Information

Each CSTA service request will allow the inclusion of information that has not been standardized. This allows private information to be passed between the application processes involved in the call. Examples of such information are authorisation codes, security information, account code or other billing information.

10.1.5 Other Parameters

The service parameter lists are not exhaustive. Other parameters are for further study.

10.2 Switching Function Services

10.2.1 Make__Call Service

The Make__Call service establishes a CSTA call between two devices. The Make__Call service establishes a CSTA call identifier that lasts for the life of that call.

10.2.1.1 Service Request

The request for the Make_Call service includes the following information.

Calling Device Identifier - this parameter indicates the static device identifier of the device from which the call is originated.

Generally, the calling device must be in either idle state (on hook) or in pending state (off hook, but not yet dialling a number). It should be valid for the calling device to be off hook and then invoke (by way of the associated data terminal) a voice terminal service, such as directory dialling.

Called Device Identifier - this parameter indicates the static device identifier of the device to be connected after the calling device has been successfully connected.

Type of Call - this information indicates the use that the client intends for the call. Examples are voice and data. Others are for further study.

In an ISDN environment where different types of devices (for example voice terminals, data terminals, etc.) attach to the Switching Function, this configuration information is not known ahead of time by the server. When such a device initiates a call request, it supplies this type information (in a protocol information element) and the server transmits it onwards for subsequent compatibility checking.

The question of checking for compatibility of higher layer capabilities and lower layer capabilities is for further study.

Non-standard Information - see "CSTA Application Private Information" (10.1.4).

10.2.1.2 Service Response

The server verifies that a service is correct and notifies the client application in order to acknowledge or reject the service request. No assumption is made about the state of the called device.

1. Acknowledgement

   If the information in the request is correct, and resources can be allocated for the operation, and the calling device is in a state that allows it to participate in the call, a CSTA call identifier is allocated by the server and then a CSTA Connection identifier is allocated by the server and returned to the client in the application acknowledgement response. The service is initiated.

   Note that the establishment of the call has not yet begun. Call progress events may be sent by the server application as the connection establishment progresses, as selected by the client application by way of the Monitor service.

2. Rejection

   See 10.1.1.2 "Operation of the CSTA Protocol" for general conditions for rejecting the request.

10.2.1.3 Service Action

The service action is to create a call between the designated devices. When the service is initiated, the calling device is prompted if necessary (implementation dependent) and when that device acknowledges (implementation dependent) a call to the called device is initiated.

The following figure illustrates the results of a Make_Call (Calling device = D1, Called device = D2). A call is established as if D1 had called D2.

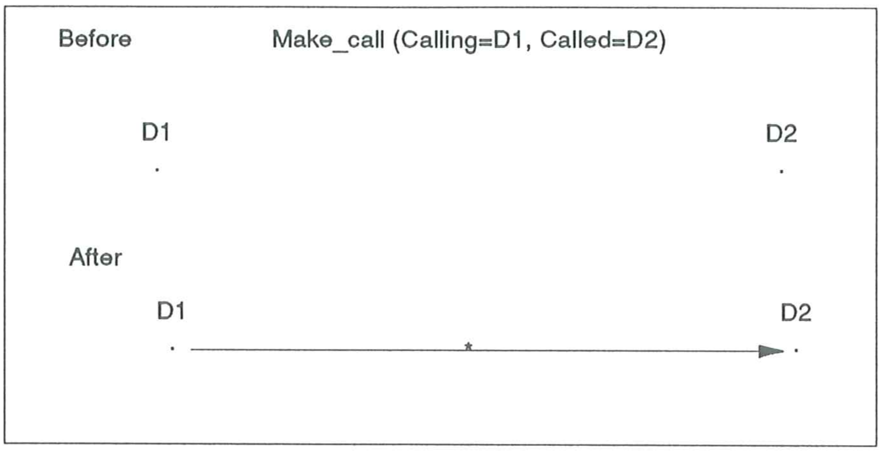


Figure 33 - Call on Hold at Device D1

10.2.1.4 Interaction with Other CSTA Conditions

To be addressed in the next stage of standards development.

10.2.1.5 Interaction with other Network Conditions

The Make__Call service acts identically to a manually-initiated call. Some of the interactions with other network conditions are:

1. Busy Condition Encountered - if the device was set to busy-forwarding, then forwarding takes place.
2. Did Not Answer Condition Encountered - if the device was set for No-Answer Forwarding, then forwarding takes place. If a No-Answer-Forwarding was not set, then the device continues to ring until disconnected by the application, or when all the existing devices have been cleared.
3. Do Not Disturb (DND) condition encountered -the service should provide the option of whether to honour DND forwarding for call initiated by a computer.

   If the device was set for DND forwarding, then forwarding takes place.

10.2.1.6 Comments

1. There are PTT requirements to limit nuisance calls to a number in a period of time. This requirement must be satisfied outside CSTA.
2. Do Not Disturb Override is provided in CSTA by the Call__Completion Service.
3. Preemption is provided in CSTA by use of the Non-Standard Information parameter.
4. Any function that can be accomplished by a user dialling directly can be provided in CSTA by the proper choice of the called party digits. Examples are:

   - Direct Trunk Select;
   - Direct Extension Select.
5. Priority service is designated by the use of the Non-Standard information parameter.

10.2.2 Clear__Call Service

The Clear__Call service releases all of the devices from the specified call. The call ceases to exist and the CSTA Call identifier is released.

10.2.2.1 Service Request

The request for the Clear__Call service includes the following information:

1. CSTA Connection Identifier:
   - Call Identifier identifies the call that is being cleared.
   - Specified device Identifier: this identifies the specified device.
2. Non-Standard Information - see "CSTA Application Private Information" (10.1.4).

10.2.2.2 Service Response

No parameters have yet been identified. The server verifies that a service is correct and notifies the client in order to acknowledge or reject the service request.

10.2.2.3 Service Action

Each device in the call is released and the connection identifier(s) (and their components) are freed. As an example, for one ISDN device of the many devices in the call, "released" means that the sequence:

- disconnect
- release
- release complete

has been completed. The following figure illustrates the results of a Clear__Call (call__id = C1, device__id = D1), where call C1 connected devices D1, D2 and D3.

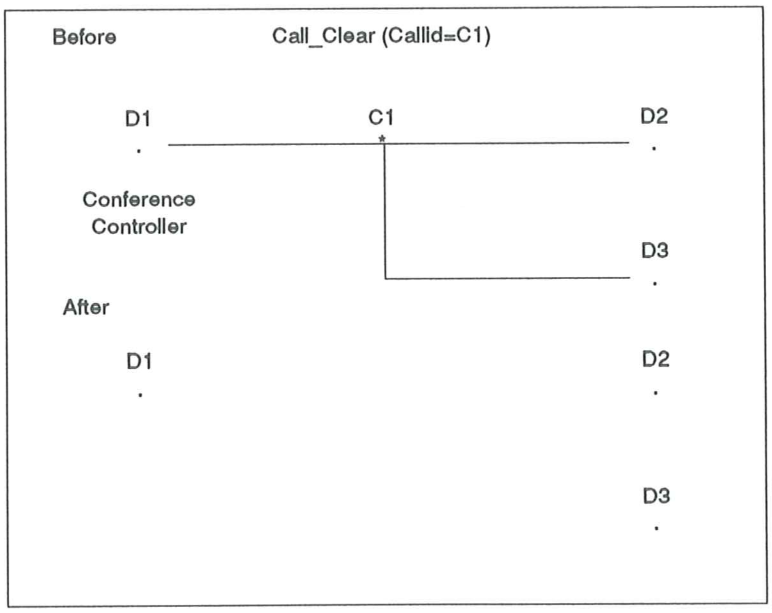


Figure 34 - Result of Successful Clear__Call Service

10.2.2.4 Interaction with other CSTA Conditions

To be supplied in the next stage of standards development.

10.2.2.5 Interaction with other Network Conditions

To be supplied in the next stage of standards development.

10.2.3 Clear_Connection Service

The Clear_Connection Service releases the specified device from the designated call. The connection is left in the Null state.

10.2.3.1 Service Request

The request for the Clear_Connection service includes the following information:

1. CSTA Connection Identifier:
   - Device Identifier - this information indicates the device that is to be disconnected.
   - Call Identifier - this information indicates the call (from among many) that the device is associated with.
2. Non-Standard Information - see "CSTA Application Private Information" (10.1.4).

10.2.3.2 Service Response

No parameters have yet been identified. The server verifies that a service is correct and notifies the client in order to acknowledge or reject the service request.

10.2.3.3 Service Action

This service releases the specified connection from the designated call.

The connection must be part of some call. As an example, for one ISDN device of the many devices in the call "release" means that the sequence

- disconnect
- release
- release complete

has been completed.

Generally, if only two connections are in the call, the effect of Clear_Connection is the same as Clear_Call.

The following figure is an example of the results of a Clear_Connection (call_id - C1, device_id = D3), where call C1 connected devices D1, D2 and D3.

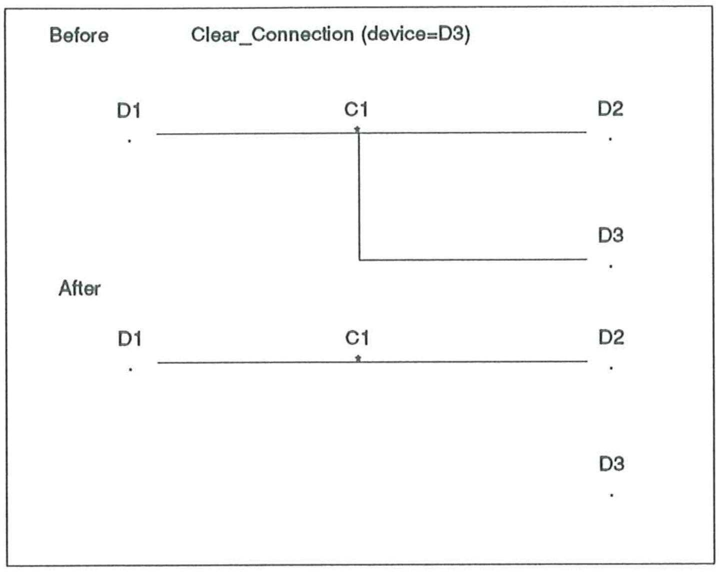


Figure 35 - Result of Successful Clear__Connection Service

10.2.3.4 Interaction with other CSTA Conditions

To be supplied in the next stage of standards development.

10.2.3.5 Interaction with other Network Conditions

To be supplied in the next stage of standards development.

10.2.4 Hold__Call Service

The Hold__Call service temporarily interrupts communication on an existing call at a device. Hold__Call acts as if the indicated device had put the call on hold. The "holding device" is the device on whose behalf the call is being placed on hold. The "held call" consists of the remaining devices in the original call.

This service is commonly known as "Consultation Hold" service in many PBXs.

10.2.4.1 Service Request

The request for Hold__Call service includes the following information:

1. CSTA Connection Identifier:
   - Device Identifier - designates the device for which hold is to be executed, i.e. the holding device.
   - Call Identifier - indicates the specific call (from among the many that the device may be involved with) for which hold is executed.
2. Connection Reservation - in an ISDN environment, a terminal requesting the hold service can optionally reserve the bearer channel for reuse by the held call. It should be possible for the CSTA service to support this option, where appropriate.
3. Non-Standard Information - see "CSTA Application Private Information" (10.1.4).

10.2.4.2 Service Response

No parameters have yet been identified.

10.2.4.3 Service Action

This service interrupts communications on an existing call at a device. A call may be placed on hold on a user's interface, by the user at any time after completion of dialling.

The associated connection is made available for other uses, depending on the reservation option. As an implementation option, the network may send a notification to the held device(s) indicating that the call has been placed on hold.

As shown in the following figure, if the Hold__Call service is invoked for device D1 in call C1, then call C1 is placed on hold at device D1. The hold relationship is remembered relative to the holding device.

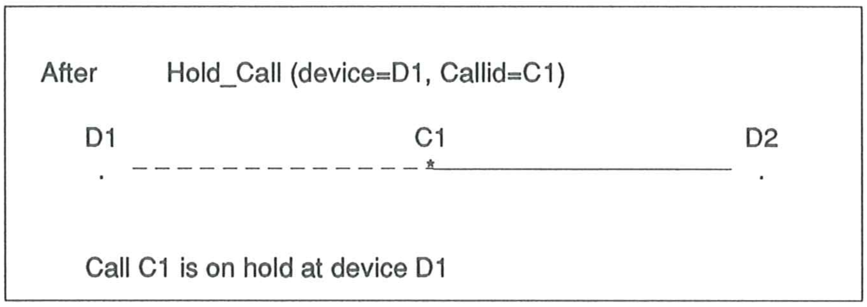

After Hold_Call (device=D1, Callid=C1)
D1
C1
D2
Call C1 is on hold at device D1

Figure 36 - Call on Hold at Device D1

Hold__Call maintains a relationship between the holding device and the held call that lasts until the call is retrieved from the hold status, or the call is cleared.

10.2.4.4 Interaction with other CSTA Conditions

To be supplied in the next stage of standards development.

10.2.4.5 Interaction with other Network Conditions

To be supplied in the next stage of standards development.

10.2.5 Retrieve__Call Service

Retrieve__Call is used to re-establish interrupted communications on an existing held call at the specified holding device.

The Hold__Call service may have reserved the held connection. The Retrieve__Call service should use the reserved connection if any.

10.2.5.1 Service Request

The request for Retrieve__Call service includes the following information:

1. CSTA Connection Identifier:
   - Device Identifier - identifies the device for which the call is to be retrieved.
   - Call Identifier - identifies the call at that device with which the operation relates.
2. Non-Standard Information - see "CSTA Application Private Information" (10.1.4).

10.2.5.2 Service Response

No parameters have yet been identified. The server verifies that the request is correct and notifies the client in order to acknowledge or reject the service request.

10.2.5.3 Service Action

The indicated device is restored into the indicated call.

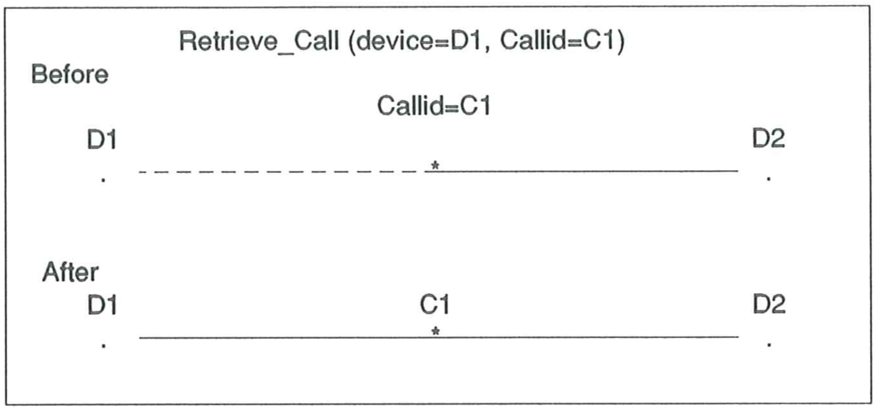


Figure 37 - Result of Successful Retrieve_Call Service

10.2.5.4 Interaction with other CSTA Conditions

To be supplied in the next stage of standards development.

10.2.5.5 Interaction with other Network Conditions

To be supplied in the next stage of standards development.

10.2.6 Consultation_Call Service

This service places an existing call on hold and makes a new call to a third device. The basic scenario is that the "served user", denoted as using device D1, wants to place his existing call to device D2 on hold and immediately make a new call to device D3.

10.2.6.1 Service Request

The request for the Consultation_Call service identifies the CSTA connection to be held (via a tuple formed from the device identifier for D1 and the associated call at device D1 which is to be held) and the consulted device identifier. The request therefore includes the following information:

1. connection to be held specified by:
   - Specified device Identifier. This identifies the user who, already having a call to device D2, wishes to make a consultation call to device D3. It is analogous to the Calling Device Identifier in the Make_Call service.
   - Existing Call Identifier; This identifies the existing call to be placed on hold prior to making the new call.
2. Consulted Device Identifier - this identifies the device to be consulted (i.e. device D3). It is analogous to the Called Device Identifier in the Make_Call service.
3. Non-Standard Information - see "CSTA Application Private Information" (10.1.4).

10.2.6.2 Service Response

On acceptance, a CSTA connection identifier is allocated for the new consultation call.

The server verifies that a service is correct and notifies the client application in order to acknowledge or reject the service request. No assumption is made about the state of the called device.

1. Acknowledgement

   If the information in the request is correct, resources can be allocated for the operation, and the calling device is in a state that allows it to participate in the new consultation call, a CSTA call identifier is allocated by the server and then a connection identifier is allocated by the server and returned to the client in the application acknowledgement response. The service is initiated.

   Note that the establishment of the new call has not yet begun. Call progress events may be sent by the server application as the connection establishment progresses, as selected by the client application by way of the Monitor service.

2. Rejection

   See 10.1.1.2 "Operation of the CSTA Protocol" for general conditions for rejecting the request.

10.2.6.3 Service Action

The service performs the compound action of placing the current, existing call on hold and then immediately initiating a second call to a third device. It may be possible to issue the Consultation_Call request with respect to a call being established (i.e. during call progress).

A Consultation_Call is similar to the Make_Call service but has some important differences. It is only valid when the user has an existing call. For this original call, the user may have been either the calling or called device (i.e. it may have been either an incoming or outgoing call). The existing call need not necessarily be in the active state; in particular it may also be possible when the user currently has a call which is already on hold.

The existing call is automatically placed on hold (if necessary) prior to the new call.

The operation of Consultation_Call is depicted in the following figure.

A rejected request leaves the existing call in its original state.

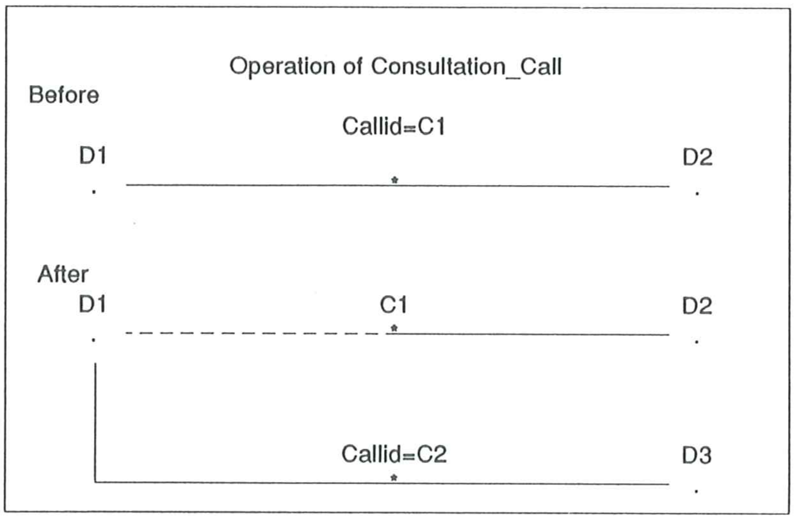


Figure 38 - Operation of Consultation_Call

10.2.6.4 Interaction with other CSTA Conditions

Make__Call Service: the consultation call could be effected by the Hold__Call and Make__Call service.

Clear__Call/Clear__Connection: the held call (to device D2) remains held.

Alternate/Reconnect__Call: these two services are used to manipulate the two calls - the held call between device D1 and D2; the consultation call between device D1 and D3.

Hold__Call/Retrieve__Call Service: the two associated calls may also be manipulated by the Hold and Retrieve operations described in the Hold service.

Transfer__Call: the Transfer service may be used after establishing the consultation call (as described above) to effect a two device call between device D2 and D3.

Conference__Call: the Conference service may be used after establishing the consultation call (as described above) to effect a three device conference call involving D1, D2 and D3.

10.2.6.5 Interaction with other Network Conditions

To be supplied in the next stage of standards development.

10.2.6.6 Application Examples

Clause 6 on Application Examples describes a type of application, called Customer Support Environment, where the agent is presented with a compound transaction formed from a voice call and an associated computer terminal screen. In handling this combined voice and data call, the agent often wants to establish a new call to involve a third device (typically for consulting another agent or an expert, etc.) while placing the original call on hold. The CSTA application will invoke this service and typically will simultaneously update the computer terminal screen of the consulted device. Consultation__Call service is aimed at this type of application.

10.2.7 Alternate__Call Service

This service provides a compound action, that causes the active call of the specified device to be held, followed by the retrieval of a previously held call. For this service to be valid, it is necessary that the specified device have at least one active call and at least one held call.

10.2.7.1 Service Request

These descriptions use the terminology adopted in the description of Consultation__Call, where the specified device, device D1, has an active call to device D3 and a held call involving device D2.

The request for the Alternate__Call service identifies the two connections to be alternated. The actual device in both connections is the same, but provision is made for identifying it twice to accommodate the use of a dynamic device identifier (which may be different in the context of each call).

Consequently, the service request includes the following information:

Active connection specified by:

- Specified Device Identifier - this identifies the user which has an active call to device D3 and a held call to device D2.
- Active Call Identifier - this identifies the active call (i.e. involving device D3) to be alternated. The information is always required when the specified device can support multiple active calls. It is not required when the specified device can only support a single active call.

Holding connection specified by:

- Holding Device Identifier - this identifies the user (as in the Active connection above). It is normally only required when using dynamic device identifiers.
- Held Call Identifier - this identifies the associated held call (i.e. involving device D2) to be alternated. The information is always required when the specified device can support multiple held calls. It is not required when the specified device can only support a single held call.

Non-Standard Information - see "CSTA Application Private Information" (10.1.4).

10.2.7.2 Service Response

No parameters have yet been identified.

10.2.7.3 Service Action

An accepted request causes the specified device's held and active calls to be swapped. The Alternate__Call service places the user's active call to device D3 on hold and, in an atomic action, re-establishes the call between device D1 and device D2 as the active call. In a similar way to Consultation__Call, device D3 can be considered as being automatically placed on hold immediately prior to the retrieval of the held call to device D2.

The operation of the Alternate__Call service is depicted in the following figure.

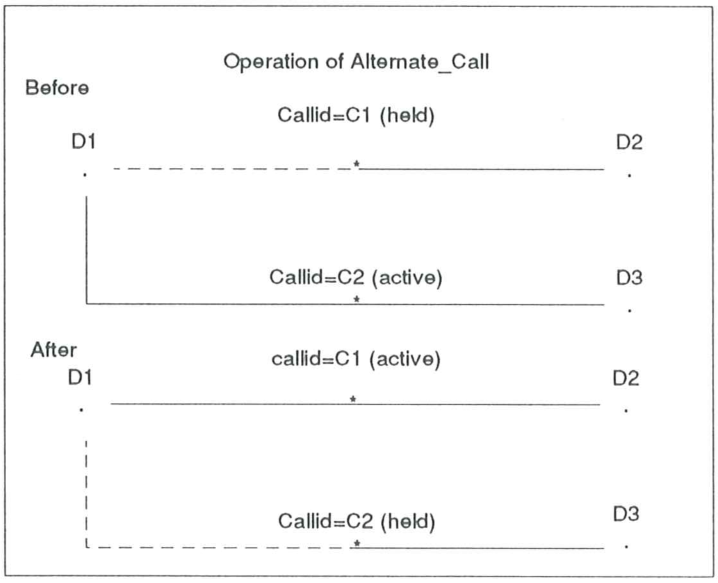


Figure 39 - Operation of Alternate__Call

10.2.7.4 Interaction with Other CSTA Conditions

Hold__Call service - the first part of this service (placing the active call on hold) could be provided by the Hold__Call service.

Retrieve__Call service - the second part of this service (restoring the held call to the active state) could be provided by the Retrieve__Call service.

10.2.7.5 Interaction with other Network Conditions

To be supplied in the next stage of standards development.

10.2.7.6 Application Examples

Clause 6 on Application Examples describes a type of application, called Customer Support Environment, where the agent is presented with a compound transaction formed from a voice call and an associated computer terminal screen. In handling this combined voice and data call, the agent often wants to establish a second telephony call to consult with a colleague. The CSTA application will invoke this call and simultaneously update the data terminal screen of the consulted colleague. A common feature in this type of application is for the original agent (i.e. the specified device) to swap between his two calls, normally maintaining privacy between the two calls. The Alternate__Call service is aimed at this type of application.

10.2.8 Reconnect__Call Service

This service provides a compound action, on behalf of the specified device, that causes the active call (in any state) of the specified device to be cleared, followed by the retrieval of a previously held call.

10.2.8.1 Service Request

These descriptions use the terminology adopted in the description of Consultation__Call, where the user, termed device D1, has a consultation call to device D3 and a held call involving device D2.

The request for the Reconnect__Call service identifies the connection to be cleared and the connection to be retrieved. Each connection is specified, in principle, by the tuple of device identifier (device D1) and the relevant call identifier (one to be cleared, one to be retrieved). The actual device in both connections is the same, but provision is made for identifying it twice to accommodate the use of a dynamic device identifier (which may be different in the context of each call).

Consequently the service request includes the following information:

Connection to be cleared specified by:

- Specified Device Identifier - this identifies the specified device which has a consultation call to device D3 and a held call to device D2. The specified device must have an existing call (either active or in some stage of call establishment) and an existing held call.
- Call Identifier - this identifies the call to be cleared.

Connection to be retrieved specified by:

- Device Identifier - this identifies the user (as in the connection above). It is normally only required when using dynamic device identifiers.
- Call Identifier - this identifies the associated held call to be reconnected.

Non-Standard Information - see "CSTA Application Private Information" (10.1.4).

10.2.8.2 Service Response

No parameters have yet been identified.

10.2.8.3 Service Action

An accepted request causes the existing call to be cleared. Having cleared the call, the held call involving device D2 is retrieved and becomes active. This service will typically be used to clear an active call and return to a held call; however, it can also be used to effect a cancel of a consultation call (because of no answer, called device busy, etc.) followed by returning to a held call.

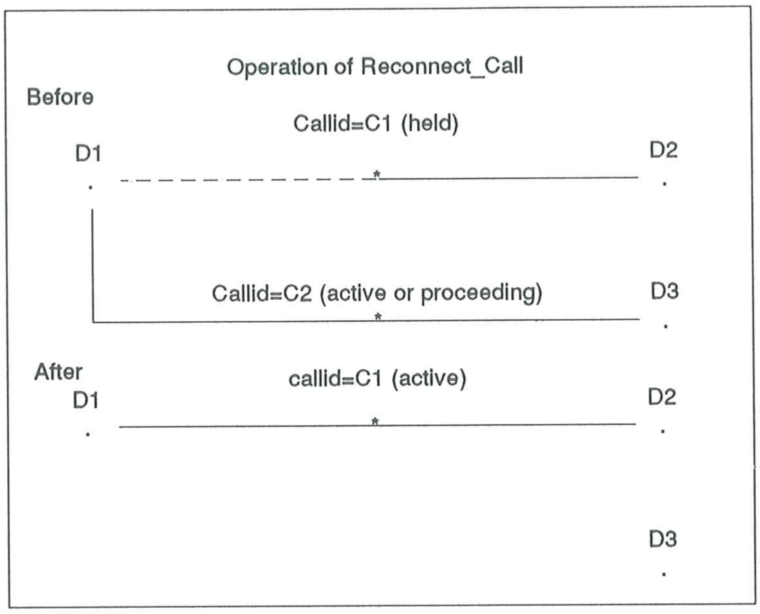


Figure 40 - Operation of Reconnect__Call

10.2.8.4 Interaction with other CSTA Conditions

Clear__Call Service - the initial clearing of the active call may be achievable by the Clear__Call service.

10.2.8.5 Interaction with other Network Conditions

To be supplied in the next stage of standards development.

10.2.8.6 Application Examples

Clause 6 on Application Examples describes a type of application, called Customer Support Environment, where the agent is presented with a compound transaction formed from a voice call and an associated computer terminal screen. In handling this combined voice and data call, the agent often wants to establish a second telephony call to consult with a colleague. The CSTA application will invoke this call and simultaneously update the data terminal screen of the consulted colleague. A common feature in this type of application is for the original agent (i.e. the user) to want to release the consultation call and return to the original held call. The Reconnect__Call service is aimed at this type of application.

10.2.9 Transfer__Call Service

This service initiates the transfer of an existing held call between devices D1 and D2 into a call from device D2 to a device D3 which was previously consulted by device D1.

10.2.9.1 Service Request

The request for the Transfer__Call service includes the following information:

Held Connection

- Specified Device Identifier 1: this identifies the specified device D1, which has already a held call to D2 and which wants to make a transfer of this call.
- Call Identifier: designates the 2- device call D1 to D2 which has to be transferred.

Active Connection

- Specified device Identifier 2: this identifies the specified device D1, which has already a consultation call to D3.
- Call Identifier: designates the consultation call from D1 to D3, which was initiated from device D1, or which is active at device D1.

Non Standard Information - see "CSTA Application Private Information" (10.1.4).

10.2.9.2 Service Response

If the information in the request is correct and resources can be allocated for the operation, then a CSTA Connection__id for the new call D2 to D3 is allocated and the service is initiated.

Return parameters:

1. Connection__id of call
2. Device__id of device

This Connection__id is a Connection__id for any device in the new call.

The return of these parameters is not always possible or does not always make sense: if the call is to be transferred out of the sub-domain, the Connection__id cannot be returned. Or, because D1, after the transfer is no longer involved in the transferred call, the application may not be interested in the parameters. Generally, the CSTA reporting mechanisms, e.g. the Monitor service, can be used to acquire the needed information on the results of the performed services. This is for further study.

10.2.9.3 Service Action

If the service request was rejected, then there is no service action. If the service request was acknowledged, then the following actions are taken.

The starting conditions are: the call C1 from D1 to D2 is in held state. A call C2 from D1 to D3 is in progress.

If the request is used in the situation where the call from D1 to D3 is established or if the call is in any state other than error/null state then the transfer can be completed.

If the transfer service successfully completes, then D1 is released from the call.

10.2.9.4 Interaction with other CSTA Conditions

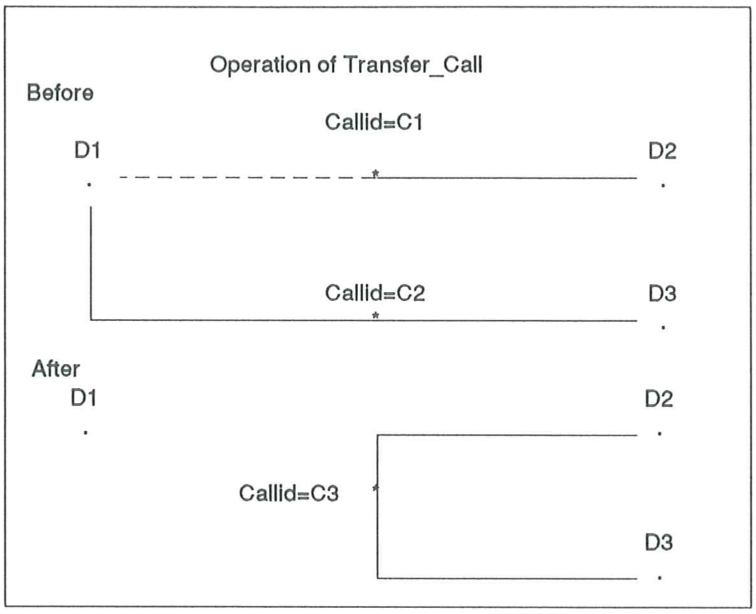


Figure 41 - Operation of Transfer__Call

The CSTA Reconnect__Service can be used to clear the call C2 (in any state) and to reconnect D1 to D2.

10.2.9.5 Interaction with other Network Conditions

To be supplied in the next stage of standards development.

10.2.9.6 Application Examples

Clause 6 on application descriptions contains the description of the Customer Support Environment application. In this application the agent often has established a second call to a third device (e.g. having a consultation call to another person). Sometimes the call has to be transferred to the third person, which continues the customer transaction and device D1 leaves the call. This situation can be handled by the CSTA application issuing the Transfer__Call service request.

10.2.10 Conference__Call Service

This service initiates the conference with an existing held call between devices D1 and D2 and a call from device D1 to a device D3 which was previously consulted by device D1.

10.2.10.1 Service Request

The request for the Conference__Call service includes the following information:

Held Connection

Device Identifier 1: this identifies the specified device D1, which has a held call to D2 and which wants to make a conference with this call.

Call Identifier: designates the 2- device call D1 to D2 which has to be conferenced.

Active Connection

Device Identifier 2: this identifies the device D1, which has a consultation call to D3.

Call Identifier: designates the consultation call from D1 to D3, which was initiated from device D1, or which is active at device D1.

Non Standard Information - see "CSTA Application Private Information" (10.1.4).

10.2.10.2 Service Response

If the information in the request is valid and resources can be allocated for the operation, then a CSTA connection ID for the new call involving D1, D2 and D3 is allocated and the service is initiated.

Return parameters:

- Connection_id of call
  . Device_id
  . Call_id

This connection_id is a connection_id for any device in the new call.

The return of these parameters is not always possible or does not always make sense. Generally, the CSTA reporting mechanisms, e.g. the Monitor service, can be used to acquire the needed information on the results of the performed services. This is for further study.

10.2.10.3 Service Action

If the service request was rejected, then there is no service action. If the service request was acknowledged, then the following actions are taken.

The starting conditions are (following figure): the call C1 from D1 to D2 is in held state. A call C2 from D1 to D3 is in progress.

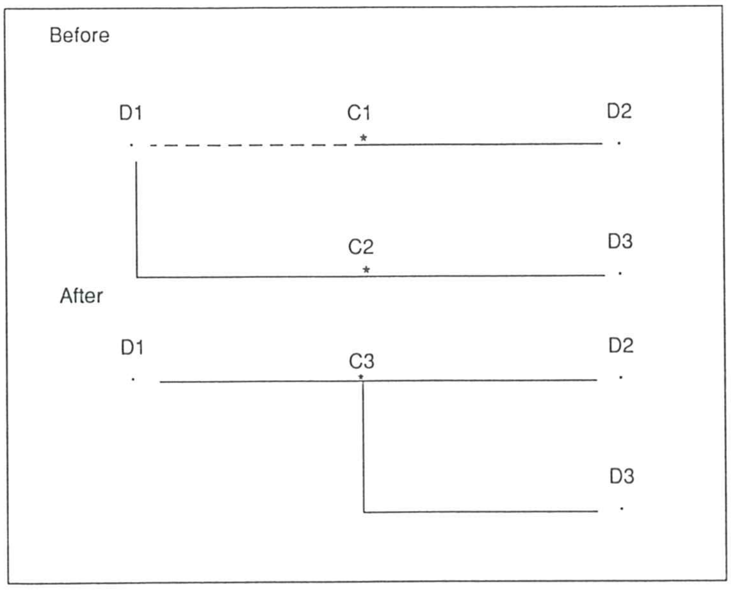


Figure 42 - Operation of Conference_Call

If the request can be used in the situation where the call from D1 to D3 is established then the conference can be completed.

If the Conference service successfully completes, D1, D2 and D3 are conferenced together.

10.2.10.4 Interaction with other CSTA Conditions

The CSTA Reconnect__Service can be used to clear the call C2 and to reconnect D1 to D2 before the conference is complete.

10.2.10.5 Interaction with other Network Conditions

To be supplied in the next stage of standards development.

10.2.11 Answer__Call Service

The Answer__Call service answers the specified call on the specified device. For example, Answer__Call activates the speaker or headset at a phone, thus allowing hands-free operation of the phone.

The device should be one that can be auto-answered by an application.

10.2.11.1 Service Request

The request for Answer__Call service includes the following information:

Call to be answered

Call Identifier -indicates the call to which the operation applies.

Device Identifier - indicates the device to which the connection is to be activated.

Non-Standard Information - see "CSTA Application Private Information" (10.1.4).

10.2.11.2 Service Response

No parameters have currently been identified.

10.2.11.3 Service Action

The Answer__Call is expected to occur after the incoming call has been presented to the user.

10.2.11.4 Interaction with other CSTA Services

To be supplied in the next stage of standards development.

10.2.11.5 Interaction with other Network Services

To be supplied in the next stage of standards development.

10.2.11.6 Application Examples

The client may wish to request that a voice terminal be answered on behalf of a party who is physically impaired or for ACD agents equipped with headsets but not handsets.

10.2.12 Call__Completion Service

The Call__Completion Service provides a mechanism to invoke features that complete a call which may otherwise fail. Generally these services are invoked when a call is set up and encounters a busy far device or no answer.

10.2.12.1 Service Request

The service request should include the following elements:

Connection Identifier

call__id - indicates the call on which the feature is requested.

device__id - indicates the device for which the feature is requested.

Feature - identifies the feature to invoke on the call. Possible features include:

- Camp On - allows queueing for availability. Generally this service has the caller wait until the called party finishes the current call and any previously camped on calls.
- Call Back - allows requesting the called device to return the call when it returns to idle. This service works much like Camp On, but the caller is allowed to hang up after invoking the service, and the CSTA Switching Function calls both parties when the called party becomes free.
- Intrude - allows the caller to be added into an existing call.

Non Standard Information - see "CSTA Application Private Information" (10.1.4).

10.2.12.2 Service Response

No parameters have currently been identified.

10.2.12.3 Interaction with other CSTA Conditions

To be supplied in the next stage of standards development

10.2.12.4 Interaction with other Network Conditions

To be supplied in the next stage of standards development.

10.2.13 Make__Predictive__Call Service

The Make__Predictive__Call service establishes a basic CSTA call between two devices. The results of the Make__Predictive__Call service and the Make__Call service are very similar in that they connect the calling device and the called device. They differ, however, in the order that the devices are joined to the call. Make__Call begins by joining the calling devices to the call, while Make__Predictive__Call begins by joining the called device to the call.

The Make__Predictive__Call service establishes a CSTA call identifier that lasts for the life of that call.

This service will often be used to initiate a call from a group of devices (or a logical device). This service will allocate the call to a particular device within that group at some time during the progress of the call.

10.2.13.1 Service Request

The request for the Make__Predictive__Call service includes the following information:

Calling Device Identifier - this parameter indicates the static device identifier of the device from which the call is originated.

Generally, the calling device must be in either idle state (on hook) or in pending state (off hook, but not yet dialling a number). It should be valid for the calling device to be off hook and then invoke (by way of the associated data terminal) a voice terminal service, such as directory dialling.

Called Device Identifier - this parameter indicates the static device identifier of the device to be connected after the calling device has been successfully connected.

Type of Call - this information indicates the use that the client intends for the call. Examples are voice and data. Others are for further study.

In an ISDN environment where different types of devices (for example voice terminals, data terminals, etc.) attach to the Switching Function, this configuration information is not known ahead of time by the server. When such a device initiates a call request, it supplies this type information (in a protocol information element) and the server transmits it onwards for subsequent compatibility checking.

The question of checking for compatibility of higher layer capabilities and lower layer capabilities is for further study.

Non-standard Information - see "CSTA Application Private Information" (10.1.4).

Allocation - this information specifies the condition when the outbound call is allocated to the client:

- Call__Delivered: this means that the called number is at least not busy and may eventually be answered. For some applications, an agent can be allocated to anticipate the answer.
- Call__Established: this means that the called number has actually answered.

NOTE 4
If the Computing Function wishes to allocate the call to a specific agent, then the Make__Call service in conjunction with the monitor service, hold service and transfer service may be used.

10.2.13.2 Service Response

The server verifies that the request is correct and notifies the client application in order to acknowledge or reject the service request. No assumption is made about the state of the called device.

1. Acknowledgement

   If the information in the request is correct, resources can be allocated for the operation, and the calling device is in a state that allows it to participate in the call, a CSTA call identifier is allocated by the server and then a CSTA Connection identifier is allocated by the server and returned to the client in the application acknowledgement response. The service is initiated.

   Note that the establishment of the call has not yet begun. Call progress events may be sent by the server application as the connection establishment progresses, as selected by the client application by way of the Monitor service.

2. Rejection

   See 10.1.1.2 "Operation of the CSTA Protocol" for general conditions for rejecting the request.

10.2.13.3 Service Action

The service first initiates a call to the called device. Depending on the Allocation parameter, the call is allocated to an actual physical device during the progress of the call. The allocated device will be prompted (implementation dependent) if necessary.

The following figure illustrates the results of a Make__Predictive__Call (Calling device = group containing device D1, Called device = D2, Allocation = Call__Established).

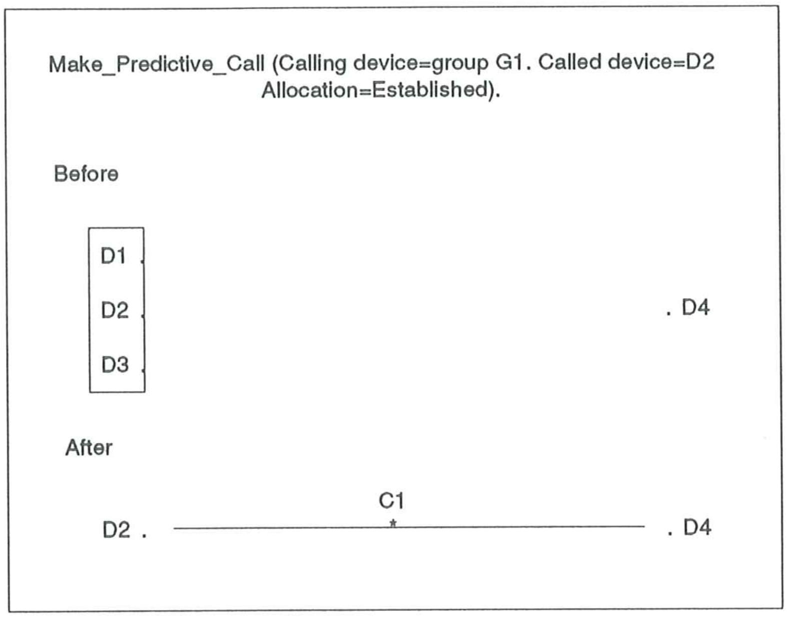

Make_Predictive_Call (Calling device=group G1. Called device=D2
Allocation=Established).
Before
D1
D2
D3
D4
After
C1
D2
D4

Figure 43 - Operation of Make__Predictive__Call

10.2.13.4 Interaction with Other CSTA Conditions

To be addressed in the next phase of standards development.

10.2.13.5 Interaction with other Network Conditions

1. Busy Condition Encountered - if the device was set to busy-forwarding, then forwarding takes place.
2. Did Not Answer Condition Encountered - if the device was set for No-Answer-Forwarding, then forwarding takes place. If No-Answer Forwarding was not set, then the device continues to ring until disconnected by the application, or when all the existing devices have been cleared.
3. Do Not Disturb (DND) Condition Encountered -If the device was set for DND forwarding, then forwarding takes place.

10.2.14 Divert__Call Service

The Divert__Call service changes the destination of an incoming call from one device to another. The Divert service can complete successfully only if the specified connection is in the alerting state.

10.2.14.1 Service Request

The request for Divert service includes the following information:

Connection Identifier:

- Call Identifier - contains the identifier of the call to which the operation applies.
- Called Device Identifier - contains the identifier of the original called device, that is, the device from which the call is being diverted.

New Device Identifier - indicates the static device identifier of the device that is to become the new called device as a result of this service.

Non-Standard Information - see "CSTA Application Private Information" (10.1.4).

10.2.14.2 Service Response

No parameters have yet been identified.

10.2.14.3 Service Action

The Divert service replaces the original called device, as specified in the Called Device Identifier, with a different called device, as specified in the New Device Identifier.

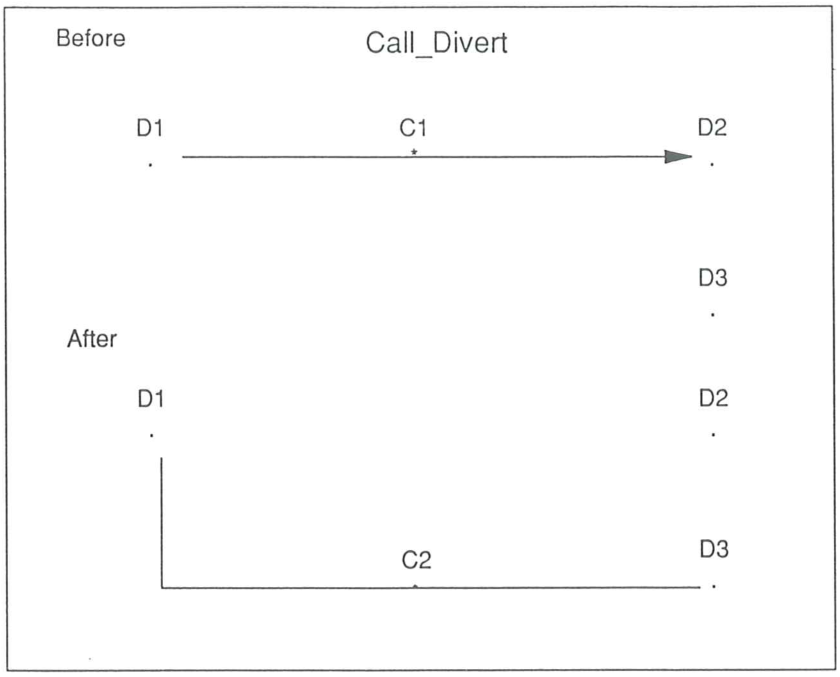


Figure 44 - Operation of Divert__Call

10.2.14.4 Interaction with Other CSTA Conditions

To be addressed in the next phase of standards development.

10.2.14.5 Interaction with other Network Conditions

To be addressed in the next phase of standards development.

10.2.15 Feature Access Service

The Feature Access Service provides a query mechanism to determine the state of device features, like do not disturb and message waiting indicator, and a request mechanism to set those features as well. Currently the service is focused on those features that are accessible to a user of the Switching Function. Administration is outside the scope of CSTA.

10.2.15.1 Service Request

The service request should include the following elements:

static device__id: indicates the telecommunications endpoint to which the query/set applies.

set/query: flag that indicates whether to set or query the feature.

NOTE 5
The features that can be set and queried, is for further study.

feature: identifies the requested feature. Possible features accessed in this manner may include:

- message waiting indicator indicates messages available
- do not disturb indicates that the device is in the do not disturb mode
- forward indicates whether the device is forwarding calls along with the type of forwarding and the forwarding number
- last number indicates the last number dialled.
- device type indicates the type of device. This can include one or more of the following attributes e.g.:
    . Voice
    . Hands free
    . Data
    . ACD
    . Group
    . Other
- Agent State indicates ACD agent state if the device is part of an ACD group. This state may include the following values:
    . On Call
    . Idle
    . After Call Work
    . Logged Out
    . Other Work

Non-Standard Information - see "CSTA Application Private Information" (10.1.4).

10.2.15.2 Service Response

The service response will simply acknowledge or refuse requests to set features. Non-refused requests to query feature states will also include the requested feature state as part of the response.

10.2.15.3 Interaction with Other CSTA Conditions

To be addressed in the next phase of standards development.

10.2.15.4 Interaction with other Network Conditions

To be addressed in the next phase of standards development.

10.3 Input/Output Services

A mature Input/Output Services description will be provided in Edition 2. The following material is a collection of preliminary information from discussions that have taken place up to now.

The Input/Output services allow the CSTA application to use terminals of the switching system (telephone, card reader etc) for data entry and display.

Applications for such services are described in Clause 6.

To initiate and terminate the Collect__Data service from the computer, the Collect__Data service request is used. To initiate and terminate the service from the switch the Start-Data-Collection service is used.

The actual input data is transmitted from the switching system to the computing system via the Input__Data notification (event) (see following figure). Input is solicited from the user by writing text and prompts on the set's display via the Provide__Data notification (event).

A speech path may be used concurrently with input/output services. In order to accommodate interactions with associated and non-associated speech paths and with other data paths, the switch may be required to dynamically alter characteristics of the data path. The Activate__Data__Path notification (event) is used for this purpose.

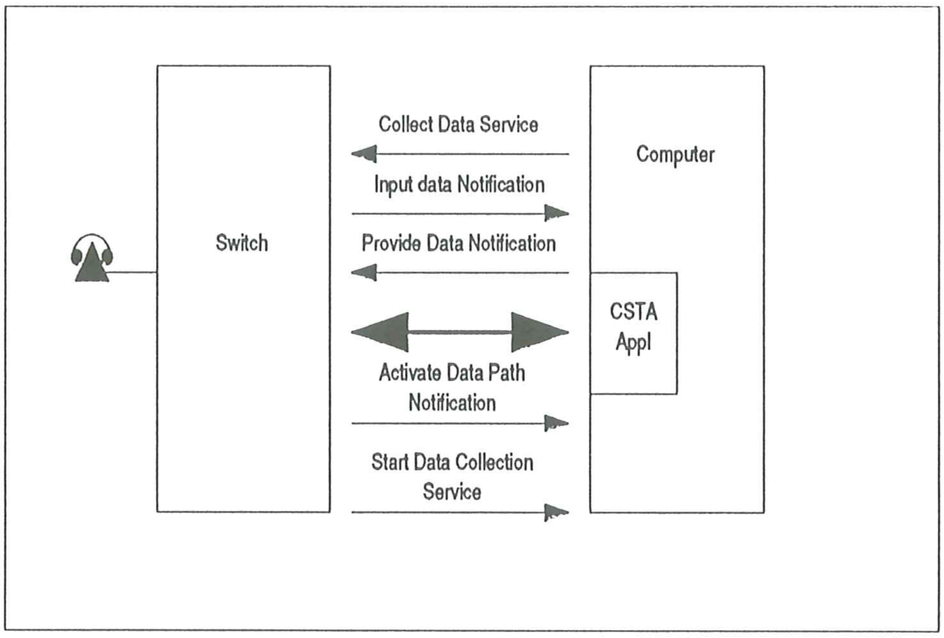

Switch
Collect Data Service
Input data Notification
Provide Data Notification
Activate Data Path Notification
Start Data Collection Service
Computer
CSTA Appl

Figure 45 - Operation of Input/Output Service

10.3.1 Collect__Data service

10.3.1.1 Service Request

This request is sent from the computer application to the switch. It informs the switch that data input from a specified terminal is to be sent to the application (via Input__Data event), and that the application may subsequently use Provide-Data to write to the device.

Device:

It designates the device from which the data is to be collected.

Start/Stop Data Entry:

It specifies whether data collection is to be initiated or terminated.

Additional Control Information:

It is used to specify additional control information sent to the switch which generates actions at the designated voice terminal in preparation of the data entry e.g.

- giving audible tones;
- displaying a message

Call:

If the application is requesting an association between a voice path (a call) and a data path (the device's video display screen), then this parameter will indicate the voice connection with which to establish the association.

Thus if the user places a call with a data path association on hold and the call is subsequently retrieved from hold; the switch will be able to inform the computer application that the device's screen is once again available.

10.3.1.2 Service Response

The request has to be acknowledged to inform the application of the proper initiation or termination of the service.

The service response also returns the following information:

Device Type:

This information designates the type of device upon which the collect__data service is being initiated. Potential devices are:

- analogue voice terminal (sub-types possible);
- digital voice terminal (sub-types possible);
- analogue trunk;
- digital trunk.

The computer application can thereby be informed if the terminal from which the service is being used has for example, a character display for visible output of data and the dimensions of that display if one is present.

Currently Active Indicator

This information field, in conjunction with the activate__data__path event, provides a mechanism for synchronizing the data path with the voice path.

This information indicates whether the Collect__Data service is currently active or suspended. A Collect__Data service may be established in a suspended mode if a voice path association was specified and that voice path is itself currently inactive (on hold).

The switch may deactivate the data path as a by-product of the user making a call (via the Make__Call service or via the direct action on the set itself) or by placing the data path or associated voice path on hold.

10.3.1.3 Service Action

If the service request was rejected, there is no action.

If the service request was accepted, the switch is ready to receive and collect data entered at the voice terminal. Depending on the Additional Control Information (see above), specific actions may be taken to give the user at the Switching Function terminal some guidance. Information collected would be sent to the computer. (See "Data__Input Notification" below). The switch is also ready to display data supplied by Provide__Data (see "Provide__Data notification" below).

10.3.1.4 Interaction with other CSTA Conditions

Make__Call Service: The invocation of the service can occur independent of a call. If the service is activated, and subsequently a call to a device D2 has been made from the telephone (device D1) where the data was being entered, then the Collect Data Service will be parked ("Suspended"). No data can be entered during this phase. The logical connection between the switch and the application will not be cleared so that data collection can resume after the call.

Hold__Call Service, Retrieve__Call Service: The invocation of the service can also be synchronized with a call. If the service is activated with an associated call, and subsequently the call is placed on hold, the Collect Data Service will be parked ("Suspended"). No data can be entered during this phase. The logical connection between the switch, the application, and the call will not be cleared so that data collection can resume after the call is retrieved.

10.3.1.5 Interaction with other Network Conditions

To be addressed in the next phase of standards development.

10.3.2 Data__Input Notification

This notification is an event sent from the switch after the computer has requested the data entry via the Collect__Data request.

It conveys user data messages entered from a voice terminal to the CSTA application.

The notification for Data Input from the switch to the computer includes the following information.

Device: This information designates the endpoint from which the data is entered.

Call: This information designates the voice path with which the data path is associated.

Time/Date: It can be used to give the actual time and date when the data was entered at the terminal.

User Data: It contains data entered either directly by the user and/or by the peripheral and/or by the switch on the user's behalf. The data includes the following information at least one of which must be present.

String: The end of the character string is defined by an "End Of Text" character (e.g. Return). It could also be provided in a separate length parameter. In that case the switching system would be responsible for detecting the end of message and for generating the correct length indication. A single keystroke would be represented as a string of length "1".

Mobile User Number: In cases where a mobile user can be determined by the switch, by a card reading device or by another peripheral, it could be presented here. Mobile user number is applicable in the case where a 'mobile user' with a directory number different from the number at which the data was entered, is using the service.

User ID: In cases where a user ID can be determined by the switch, by a card reading device or by another peripheral, it could be presented here. The User ID information contains an identity code which is entered at the Switching Function terminal (manually or with a card reader) prior to entering the user data. The application (in the switch and/or the computer) can then check if the user is allowed to use the Collect__Data service.

10.3.2.1 Interaction with Other CSTA Conditions

To be addressed in the next phase of standards development.

10.3.2.2 Interaction with other Network Conditions

To be addressed in the next phase of standards development.

10.3.3 Provide_Data Notification

Provide_data notification is sent from the computer to the switch after the computer has requested data entry via the Collect_Data request.

It allows the CSTA application to display data (characters) and/or voice information at the Switching Function terminal. The display of data requires the availability of a display at the terminal or device. Voice information can be displayed in the form of voice prompts or tones.

The notification for provision of data from the computer to the switch includes the following information. Zero or more occurrences of each information field may be present in a single message, and these fields should be displayed (visually or acoustically) sequentially by the device.

Device: It designates the terminal to which the user data are to be transmitted.

Call: This information designates the voice path association for which the data are provided.

Voice Prompt Number: If a voice prompt is to be displayed at the designated voice terminal, then the selection is performed by using a prompt number. A voice prompt may be used with or without a text prompt.

Text Prompt: If a prompt is to be displayed on the character display, then this information can be used to control that action. A text prompt may be used with or without a voice prompt.

This information includes a prompt identifier (which will map to a vendor/device specific location), the prompt text and an indicator of whether the item is currently selected (i.e. a particular vendor may show selected items in highlight, reverse video etc).

Display Tone Control: It is used for control of the tones to be displayed. Details are for further study.

Move Cursor: It contains the row and column co-ordinates of the location at which a digit string or character string will begin.

Number of Digits: It is used to specify the number of digits to be displayed acoustically or optically at the terminal.

Digit String: It contains the sequence of digits to be displayed.

Number of Characters: It is used to specify the number of characters to be displayed at the terminal.

Characters: It contains the sequence of characters to be displayed, possibly including new lines.

Clear Display: It is used to clear the device's entire display.

Clear to End of Line: It is used to clear from the current cursor position to the end of the current row.

Local Echo: It is used to indicate that characters entered at the device should be locally echoed either as text beginning at the current cursor location and/or over the audio path as tones.

10.3.3.1 Interaction with Other CSTA Conditions

To be addressed in the next phase of standards development.

10.3.3.2 Interaction with other Network Conditions

To be addressed in the next phase of standards development.

10.3.4 Activate__Data__Path notification

Activate__data__path is an event sent from the computer to the switch after the computer has requested data entry via the Collect__data request.

It informs the computer based CSTA application of changes in data path's characteristics. The event is used to synchronise the data path availability with voice path(s).

The event is initiated by the switch to indicate that the application's data path has been temporarily deactivated ("suspended", "parked"). This may be as a consequence of placing a call (either via the Make__Call service or by direct action on the set) or as a consequence of the user placing the data path or the associated voice path on hold.

The event is also initiated by the switch to indicate that the application's data channel has been re-established ("retrieved"). This may be a consequence of the invocation of a Retrieve__Call service or of direct user action on the set.

The event is initiated by the computer to "request" the allocation of scarce physical devices such as tone receivers and generators in the event that the device is a trunk. The switch may refuse this request (and so inform the computer).

The Activate__Data__Path event does not involve the overhead of a device/call monitor being either established on the switch or understood by the computer application.

The requirement for such an event is described in clause 6 (Data Collection/Distribution).

The Activate__Data__Path notification from the switch to the computer includes the following information:

Device: It designates the terminal to which the user data are to be transmitted.

Call: This information designates the voice path association for which the data are provided.

Active: It indicates whether the data path has been activated or deactivated. The switch is not required to maintain an image of the data display. It is therefore incumbent upon the computer application to re-draw the display after being informed of its reactivation.

Trunk Resources: It indicates the physical resources requested by the computer in the event that the device is a trunk. These resources may include a tone generator and/or a tone receiver.

10.3.4.1 Interaction with Other CSTA Conditions

To be addressed in the next phase of standards development.

10.3.4.2 Interaction with other Network Conditions

To be addressed in the next phase of standards development.

10.3.5 Start__Data__Collection service

The Start__Data__Collection service informs the computer that a CSTA Collect__Data service has been requested by a switching system terminal (voice terminal, card reader, etc).

This service functions as a switch invoked version of the Collect__Data service and shares the data__input, provide__data and Activate__Data__Path notification (events) of the Collect__Data service.

The requirement for this service stems from the requirement that Input/output services should not require dedicated devices. Examples are presented in Clause 6 (Data Collection/Distribution).

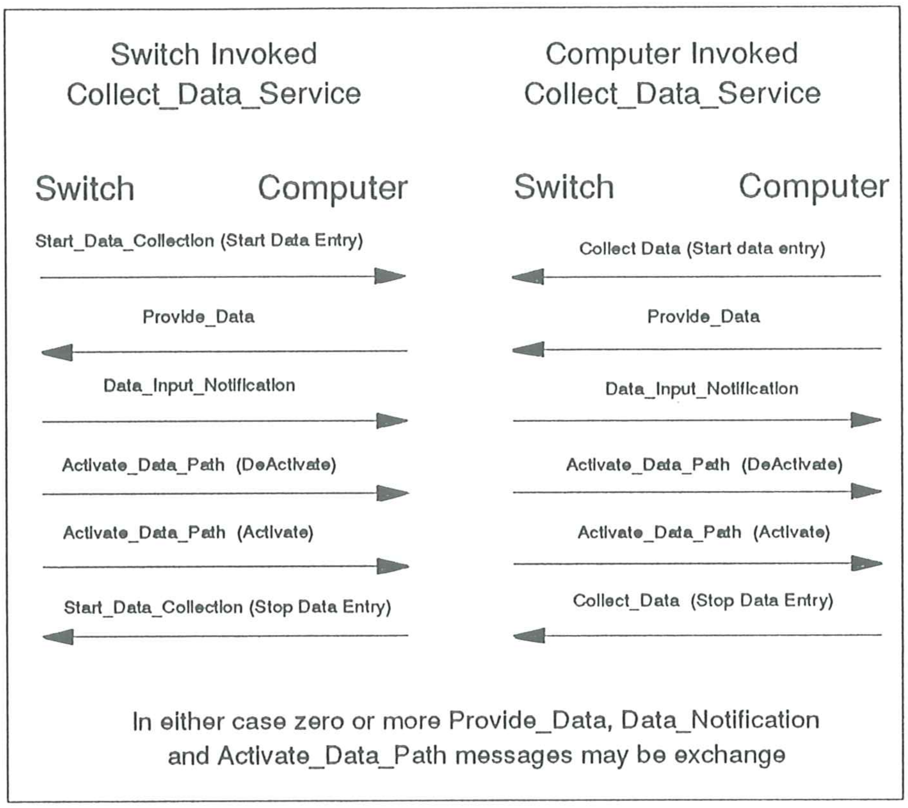

Switch Invoked Collect_Data_Service
Computer Invoked Collect_Data_Service
Switch
Computer
Switch
Computer
Start_Data_Collection (Start Data Entry)
Provide_Data
Data_Input_Notification
Activate_Data_Path (DeActivate)
Activate_Data_Path (Activate)
Start_Data_Collection (Stop Data Entry)
Collect Data (Start data entry)
Provide_Data
Data_Input_Notification
Activate_Data_Path (DeActivate)
Activate_Data_Path (Activate)
Collect_Data (Stop Data Entry)
In either case zero or more Provide_Data, Data_Notification and Activate_Data_Path messages may be exchange

Figure 46 - Message Flow Examples

10.3.5.1 Service Request

The request for Start__Data__Collection from the switch to the computer includes the following information.

Device: It designates the terminal which is requesting a collect__data service.

Call: This information designates the voice path association for the collect__data service's use. Both the switch and the computer must have a-priori knowledge whether or not a call path is required for the commencement of this particular collect__data service. This would be established by system administration functions in both environments.

Service__id: It indicates the particular Collect__Data service which is requested. Both the switch and the computer must have a-priori knowledge of the meaning of particular values of the Service__id. This would be established by system administration functions in both environments.

Currently Active Indicator: It indicates whether the data path is active or inactive.

Trunk Resources: It indicates the presence of physical resources in the event that the device is a trunk. These resources may include a tone generator and/or a tone receiver. Once again both the switch and the computer must have prior knowledge of the meaning of particular values of the Service__id. This would be established by system administration functions in both environments.

Device Type: This information designates the type of device which requested the Collect__Data service. As for the Collect__Data service, the potential devices are:

- analogue voice terminal (sub-types possible);
- digital voice terminal (sub-types possible);
- analogue trunk;
- digital trunk.

The computer application can thereby be informed if the terminal from which the service is being used has for example, a character display for visible output of data and the dimensions of that display if one is present.

10.3.5.2 Service Response

The computer should acknowledge the message and begin the proper Collect__Data service using the Data__Input notification and Provide__Data events. The Collect__Data service should subsequently be terminated by using a stop data entry request from the Collect__Data service.

10.3.5.3 Service Action

If the service was acknowledged, the service is (depending on the context of the request):

1. to allocate the tone receiver and/or tone generator;
2. to re-draw the display (if the service was invoked by the switch).

10.3.5.4 Interaction with Other CSTA Conditions

To be addressed in the next phase of standards development.

10.3.5.5 Interaction with other Network Conditions

To be addressed in the next phase of standards development.

10.4 Computing System Services

Mature Computing System Service descriptions will be completed in edition 2 of the Technical Report. The following material is a collection of preliminary information from discussions that have taken place up to now.

The computing capabilities provided to the Switching Function includes the following:

1. Invoking computer application software. This allows the Switching Function to invoke by "Invoke__Application" the application to do a specific function and obtain a return result.
2. Creating and deleting files by invoking "Create__file/Delete__file".
3. Accessing (read/write) a file by using the index of the file in the file directory or by time stamp (the time of creating the file). Other possibilities are for further study.
4. Dynamically activating/deactivating the computing servers that provide the above capabilities.
5. Monitor all computer objects.

10.4.1 Database Services

10.4.1.1 Routing Service

The Routing Service provides a destination for a call to be routed when the Switching Function requests more information to route the call.

10.4.1.1.1 Service Requests and Responses

The Routing service provides the following interactive set of requests and responses:

1. Route Request
2. Re-Route Request
3. Route Select
4. Route Used
5. Route End

It is envisioned that a request for a call to be routed is made using a Route Request. This request will be satisfied with a Route Select reply. If the route provided is not usable, a Re-Route Request may be made to access alternative routes. Once routed, information detailing the final destination may be provided using a Route Used service. Finally, the Route Ended services allows ending the transaction from either function.

1. Route Request

   The Route Request is sent from the Switching Function to the Computing Function and requests a route selection for a call under control of the Switching Function. The Computing Function will provide a preferred route to use based on the input data it receives, and using the Route Select service that follows.

   Information that can be included with the Route Request includes:

   static device__id: indicates the intended destination of the call for which a route is requested.

   calling__static device__id: provides the originating number of the call.

   type: indicates the type of facilities used (i.e. ISDN bearer) for the incoming call.

   route__type: indicates the type of routing algorithm requested. This parameter may include values such as: Normal, Least Cost, Emergency, and ACD.

   priority: indicates the priority of the call, and may affect selection of alternative routes.

   setup: provides the information contained in an ISDN call setup message if available. This information may include the following:

   - Bearer Capability
   - Channel Identification
   - Network Facilities
   - Called Party Subaddress
   - Redirecting Number
   - Transit Network Selection
   - Low-Layer Compatibility
   - User-User
   - Network Specific
   - User Specific

2. Re-Route Request

   The Re-Route Request is sent from the Switching Function to the Computing Function and requests another route selection for a call under control of the Switching Function. The Computing Function will provide the next preferred route to use based on the input data it receives, and using the Route Select service that follows.

Information that can be included with the Re-Route Request is identical to that provided in the Route Request.

3. Route Select

The Route Select is sent from the Computing Function to the Switching Function and provides a route selection for a call under control of the Switching Function. The Computing Function will provide a preferred route to use based on the input data it receives from the Route Request or Re-Route Request services that preceded this one.

Information that can be included with the Route Select includes:

static device__id: indicates the endpoint to which a route is provided. The Switching Function should attempt to route the call to this endpoint.

calling__static device__id: provides the originating number of the call.

type: indicates the type of facilities used (i.e. ISDN bearer) for the outgoing call.

route__type: indicates the type of routing algorithm used. This parameter may include values such as: Normal, Least Cost, Emergency and ACD.

priority: indicates the priority of the call, and may affect further routing done by the Switching Function.

setup: provides the information to put in the ISDN call setup message. This information may have been altered from that provided in the Route Request or Re-Route Request. See the Route Request service for a list of information elements included in this parameter.

remaining__retries: indicates the number of alternative routes remaining. This element may be a non-negative integer, or may be the value 'no information' indicating that the server does not keep count, or that there is no fixed list.

feedback__request: indicates a request to receive a Route Used information element after providing the route. This can help in determining deflections or ACD-pilot-to-agent resolutions provided by the Switching Function while routing the call.

4. Route__Used

Route__Used is sent from the Switching Function to the Computing Function and provides the actual route selected for a call that has been routed using the Route__Select service. This element is optional, but can be desirable if the computing server wishes to be informed of the Switching Function resolved route. Often the route returned by the computing server will be altered by forwarding or do not disturb features, or will be resolved by an ACD from the pilot to a particular agent.

Information that can be included with Route Used includes:

static device__id: indicates the endpoint to which a route is resolved. The Switching Function has routed the call to this endpoint.

calling__static device__id: provides the originating number of the call.

type: indicates the type of facilities used (i.e. ISDN bearer) for the outgoing call.

Switching sub-domain: indicates whether the endpoint resolved to a point within the CSTA Switching sub- domain or whether the call has been routed outside the CSTA Switching sub-domain.

5. Route End

Route End is sent from either function to the paired function and signals an end to the routing interaction. It can be provided by the Switching Function when a call has been successfully routed, dropped, or when the Computing Function has failed to provide a route within a time limit. It can also be provided by the Computing Function to indicate that no (more) routes are (currently) available for the requested number.

10.4.1.1.2 Interaction with Other CSTA Conditions

To be addressed in the next phase of standards development.

10.4.1.1.3 Interaction with other Network Conditions

To be addressed in the next phase of standards development.

10.4.1.2 Database__Access Service

The Database__Access Service provides interactions between a Computing Function database and an object within the Switching Function. The need for three forms of Database__Access Service to Read, Append and Update databases has been identified.

10.4.1.2.1 Service Request

The service request can include the following elements:

object__id: indicates the telephony object which requested the computing database service

database__id: the object identifier for the database which is to provide service

service__id: identifies the database service requested. Possible services requested in this manner include:

- Database__Read allows the switching domain to read data in the computing domain.
- Database__Update allows an existing entry in a database to be amended.
- Database__Append allows new information to be added to a database.

10.4.1.2.2 Service Response

The various services will typically provide different responses. For the example services listed above, the responses could be:

- Database__Read response provides the information requested by the database query
- Database__Update response provides acknowledgement of the database update.
- Database__Append provides acknowledgement of the append.

The need to relate the Database Services to e.g. Rec. X.500,FTAM is for further study.

10.4.1.2.3 Interaction with Other CSTA Conditions

To be addressed in the next phase of standards development.

10.4.1.2.4 Interaction with other Network Conditions

To be addressed in the next phase of standards development.

10.4.1.3 Transfer__Context Service

A need has been identified for a Transfer__Context Service. This Service is not yet fully defined. A brief description is given in Appendix D.

10.5 Bidirectional Services

10.5.1 Escape Service

While most of the common switching and computing services required by CSTA are standardized, there is a requirement to be able to "escape" from standard operations in order to exploit some special feature of a manufacturer's switch or computer. A mechanism is also required to give manufacturers an opportunity to experiment with new services which may, at a later date, be standardized.

The Escape Service, described below, uses the concept of Object Identifiers, as described in ASN.1 (see CCITT Rec. X.208 and Rec. X.209, ISO 8824 and ISO 8825). Amongst other things, Object Identifiers can be used to identify a given manufacturer's equipment and services.

1. Each company and organisation, affiliated to ECMA, may apply to be allocated an Object Identifier. Further "sub-identification" of businesses and products is then left to the discretion of that company or organisation.
2. A similar mechanism may be provided for an "Escape Information Element" within other standard services. This would allow a standard service to be performed but with a manufacturer specific addition to it (for example, a Make__Call service might be made in a standard way, but with a manufacturer specific message being sent to certain sets). This subject is for further study.

10.5.1.1 Service Request

The Escape Service request contains the following information:

1. Service/Object Identifier - this will be a manufacturer/equipment/etc. identifier for the required service.
2. Data - this unrestricted set of information is necessary for the service to be carried out.

10.5.1.2 Service Response

The server function is required to "parse" the Object Identifier as far as it needs to in order to determine whether the service is relevant to it or not.

1. If the Object Identifier can be recognised as being relevant to the server network, then it will be acknowledged and processed.
2. If the Object Identifier is not relevant to the server network, it is rejected, with an appropriate cause indication.

10.5.1.3 Service Action

If the switch is able to perform the service it will do so, in its own way, using the Service Data provided. The provision of a mechanism for returning data is for further study.

10.5.1.4 Interaction with Other CSTA Conditions

If the requested service is not relevant to the server, then there will be no interaction with other (standard or non-standard) services on the server. If the request is accepted, however, then the non-standard service may well interact with other services in a manner which is beyond the scope of this Technical Report and any subsequent CSTA standard(s).

10.5.1.5 Interaction with other Network Conditions

To be addressed in the next phase of standards development.

10.6 Status Reporting Services

Status reporting is an area of CSTA services that allow following or checking of CSTA objects. These services provide no action and effect no changes in state on the objects that they operate on. They are intended to provide information necessary to synchronize co-operating applications so that real time value added services can be provided. Two services are presented here: Snapshot and Monitor.

10.6.1 Snapshot Service

The Snapshot service is intended to provide information about objects that makes further monitoring more meaningful. For example, if a CSTA application were to start working with a call or device, the events that will eventually provide synchronisation may not occur for some time. To facilitate operations before an event report synchronises the monitor, it is necessary to be able to query the current state of CSTA objects. Snapshot provides that service.

It is envisioned that the Snapshot service work for all CSTA objects. For now, this includes calls and devices. Providing this service for processes and Computing Function devices is for further study.

10.6.1.1 Service Request

The Snapshot service can be requested for either a device or a call.

If requested for a device, the request would provide:

1. static device_id: the CSTA static identifier of the device

If requested for a call, the request will provide:

1. CSTA Connection Identifier:
   - Call_id: the call identifier;
   - Device_id: the device identifier.

10.6.1.2 Service Response

The responses for the requests of a device and a call differ. In both cases however, the response is in the form of a list. Each entry in the list has three information elements.

If the request is made for a device, the response will include the device state. Each entry in the service response list will correspond to a call at the device. The service response will return the following information for each call in the list.

1. CSTA Connection Identifier:
   - Call_id: the call identifier;
   - Device_id: the device identifier.
2. Connection state.

If the request is made for a call, the response will include the call state. Each entry in the service response will correspond to a device in that call. The service response will return the following information for each device in the list.

1. Static device_id: The CSTA static identifier for the device. A null value (0) specifies that the identifier is not available.

2. CSTA Connection Identifier:
   - Call__id: the call identifier;
   - Device__id: the device identifier.
3. Connection state.

10.6.1.3 Service Action

The nature of Snapshot is to obtain status and return it in a response. This does not affect the states of any objects in the Switching Function.

In addition, it will often be useful for an application to ascertain the state of all calls (as opposed to connections) associated with a particular device. This can be considered as an additional mode of Snapshot (device) which returns the list of all calls (associated with the device) and their call states. This is a compound operation formed from a single Snapshot (device), to retrieve the connections, followed by a Snapshot (call) for each returned connection. Two modes of operation are anticipated, dependent on where this functionality is implemented.

1. The server will, on request, automatically provide this extended information as part of the Snapshot service.
2. The client will repeatedly use the basic Snapshot services to build up the complete context for the device of interest.

The justification for the first alternative is an issue for further study.

10.6.1.4 Interaction with other CSTA Conditions

Snapshot will not affect the outcome or operation of any other service. It is expected to be used in conjunction with monitoring, to provide existing state information after a monitor has begun to add meaning to further monitored event reports.

10.6.1.5 Interaction with other Network Conditions

To be addressed in the next phase of standards development.

10.6.1.6 Application Examples

Snapshot requires a-priori knowledge of either the static device__id or call__id that identifies the device or call that should report status. Because of the constant nature of static device__ids, it is assumed that a computing application will keep records of the device/static device__id mappings or will consult a Directory service that does. Call__ids are temporal, and assigned by the Switching Function, therefore a computing application must reference a prior CSTA transaction to obtain the call__id before using it as a parameter for a Snapshot of that call. The call__id can be obtained from: Make__call, Routing, Monitor, Conference and Transfer, and Snapshot on a device.

10.6.2 Monitor Service

CSTA needs the ability to monitor events that occur in the Switching Function. To achieve this, two CSTA objects can be monitored, devices and calls. Both provide the same event report messages, but may not behave the same way.

Different applications may require differing levels of notification of event reports, so monitor also includes a filtering capability, which allows the client to specify the level of notification required. (See sub-clause in 10 entitled "Control and Management of CSTA Services".)

10.6.2.1 Service Request

The object that is to be monitored should be specified in the service request. The switching model objects can be devices and calls. Computing objects may be defined by the computing models.

Devices are identified by their static device_id, but in cases where the identifier is not known, it may be necessary to identify a device using call_id and dynamic device_id. Whether providing call_id and dynamic device_id to specify a device to allow monitoring is for further study.

Calls are identified by the connection id of any connection within the call.

The initiating request must include the following:

1. Object_id: the object to start monitoring (call or device):

For switching objects:

2. CSTA Connection Identifier:
   - Call_id: the call identifier;
   - Device_id: the device identifier.
3. Filter: specifies the amount of information requested by the client. It may be as little as a single event, or as much as all available events. The exact nature of the filter mechanism is for further study.

The terminating request must include the following:

1. Object_id: the object to cease monitoring

It should be noted that only the client can issue the terminating request, but the service can also be terminated by an abort from the server.

10.6.2.2 Service Response

The service response will simply acknowledge or refuse requests.

10.6.2.3 Service Action

Once a request has been acknowledged, a set of reports describing the events that occur will be sent to the client by the server on the association that requested the monitor (See Clause 11 for examples of events). These reports will cease after the client requests termination or the server aborts. The server should abort the monitor if the object being monitored is destroyed, which can happen for call or process, or if the object leaves the CSTA domain, which can happen for all objects.

For each device monitor, all events passed by the filter will be sent for the connection state changes in all the calls that are present at the device. If a call is forwarded or transferred, the device ceases to participate in that call, and no further events are reported.

For each call monitor, all events passed by the filter will be sent for all the connection state changes that are present in the call. If a call is forwarded or transferred, devices may cease to participate in that call, but all subsequent events at the new devices are reported. It should also be noted that a call that is being monitored may have a new call_id assigned to it after a conference or transfer, and that the monitor will report the assignment of a new call_id and continue reporting events on that call.

Following the service response, event report messages will arrive. Each message will contain information requested by a monitor.

Event messages will typically indicate a new state a particular connection (or set of connections) has entered on a particular call and are described in more detail in clause 11. The information in this report can include only the connection(s) that have changed state, or the complete new call state. Which of these options to support, is a subject for further study.

It will often be useful when receiving event reports for devices to gain a fuller picture of the current state of the device. In particular:

1. The current device state (i.e. all connection states associated with the device).
2. The state of all calls currently associated with the device.

This information can be ascertained by using the Snapshot service and it is anticipated that there will be two modes of operation:

1. The Monitor service will have an option for requesting the server to retrieve this extended information and automatically provide it in every event report related to a device (thus imposing a general server overhead on all events).
2. The Client will maintain the required extended information from previous event reports and will explicitly invoke the necessary Snapshot services only when resynchronisation is desired. This will introduce extra load in the client, but may reduce server load and communications traffic.

The justification for the first alternative is an issue for further study.

10.6.2.4 Interaction with Other CSTA Conditions

Monitoring does not interact with other Switching Function services. Monitoring is required, however, to provide information that allows the proper use of other CSTA services. For instance, monitoring is required to obtain the proper information to determine when an ACD agent can be given another call via a Make__Call request.

10.6.2.5 Interaction with other Network Conditions

To be addressed in the next phase of standards development.

10.6.2.6 Application Examples

Monitoring will be used in applications that have interest only in watching particular calls or devices. It will also be used in applications that wish to operate or affect calls automatically under conditions that are determined using the monitor.

10.6.2.7 Requirements for Service

Some of the requirements for the event monitoring service that were drawn from the application scenarios in clause 6 are:

1. It should be possible to turn on and off monitoring as part of the service. It should also be possible to set a filter for the reported events so that a subset of the events are reported.
2. It should be possible to monitor different CSTA objects, like call, device, and process - including groups.
3. There should be a generic set of events which is independent of function implementation.
4. For the Switching Function, the event reports should indicate the new state entered by an endpoint in a call.
5. It should be possible to determine the CSTA objects that can be monitored.

6. It should be possible to receive events from both manually initiated and CSTA initiated calls or processes.

7. There should be an operational model that abstracts physical implementations, yet provides wide applicability. The abstraction should hide implementation differences between networks, yet provide as much of the common functionality as possible.

8. Multiple clients should be able to monitor the same CSTA object.

11. CSTA STATES AND EVENTS

11.1 Introduction

The material in this clause is included for tutorial purposes only and to reflect some potential approaches. Various examples of telephony scenarios with associated CSTA states and events are presented. Since these are only examples of an approach, it is not anticipated that the subsequent development of the protocol will necessarily conform to the particular states and events identified. The scenarios considered are not exhaustive and consequently not all potential CSTA states and events are necessarily identified. The information does show, however, the possibility of representing different environments and procedures with a generic set of states and events. The scenarios have been chosen to demonstrate some of the key CSTA services.

11.2 Goals

1. Switching Function independent: events should be independent of the Switching Function (to allow applications portability).

2. Device independent: events should, as far as possible, be independent of the actual telephony device (to present a simple model with wide applicability). In some cases, such as the distinction between "en-bloc" and "overlap sending" call establishment, this goal may not be appropriate.

3. Implementation independent: events should represent the function rather than the implementation mechanism (which will vary).

4. Application independent: events should represent the services and not how those services are used by any particular application.

5. Invocation independent: events should be independent of the way the service was requested. This means that a human user making a telephone call from a device generates the same events as the CSTA Make_Call service operating on behalf of that device.

NOTE 6
These are goals for generic events. CSTA may also be concerned with implementation-dependent events that provide additional information. This type of event information is not considered here.

11.3 Interpretation of call state

The switching model, developed in clause 9, Operational Models, provides a notation for describing connection states. It also illustrates how a set of connection states can be interpreted as a call state. Whether the mechanism to support this interpretation is a vital part of the operational model is for further study.

The diagram that follows shows an interpretation of the set of connection states as call states in a simple call scenario as call states using the same format as was used in clause 9.

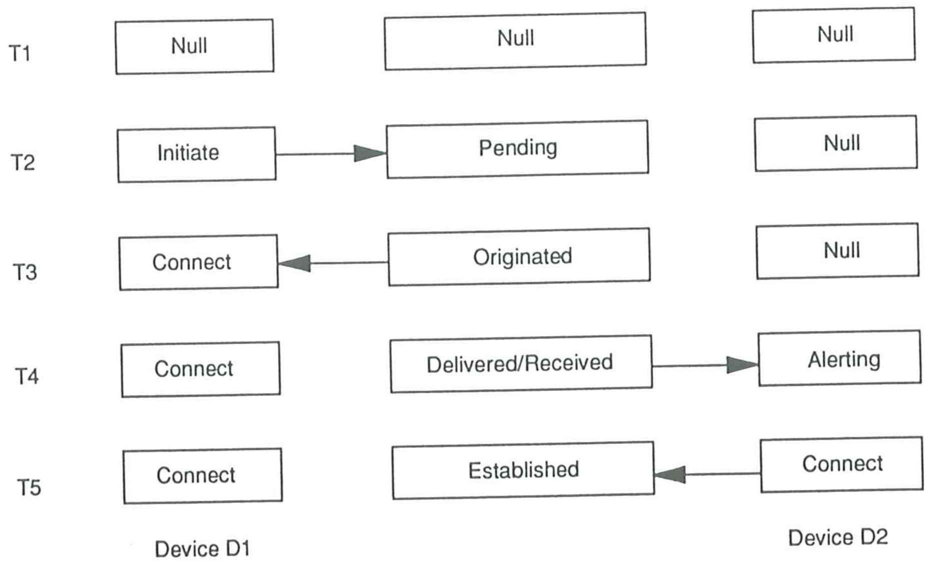


Figure 47 - Interpretation of Call State

11.4 CSTA Telecommunications Scenarios

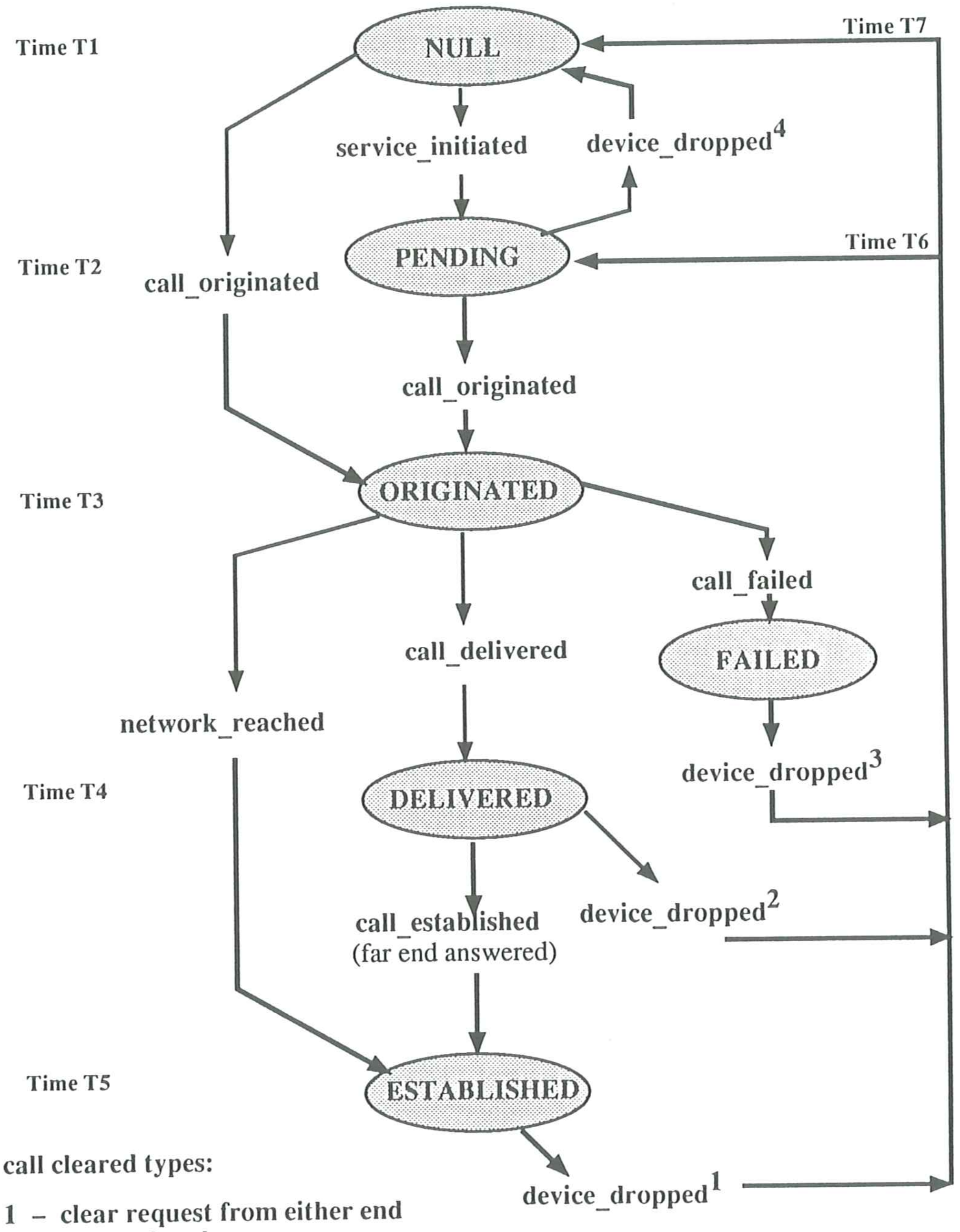
Various telecommunications scenarios are considered for different environments. The events reported by the Monitor service are an abstraction of the actual signalling employed and can be used to construct a CSTA state diagram. The intention of the following text is to investigate sets of CSTA events and states at different levels of abstraction and determine whether those events and states can be applied to a range of situations. Each scenario assumes that the necessary CSTA conditions have been established (e.g. appropriate association, monitor initialisation, etc.) and shows typical events representing the telecommunications activity.

In general, the scenarios can be considered at two levels of abstraction:

1. A level of abstraction showing how call states and events can be presented in relation to a particular device.
2. A level of abstraction showing how call states and events can be presented independently of any particular device.

11.4.1 Making and Clearing a Call

11.4.1.1 Successful Outgoing Call



call cleared types:

1 – clear request from either end
2 – caller abandons on no answer
3 – caller abandons on failure
4 – calling device goes "on hook"

Note: The annotated times relate the diagram to the following descriptions

Figure 48 - Making and Clearing a Call

Call C1 at time:

Devices		t1	t2	t3	t4	t5	t6	t7
	D1	N	I	C	C	C	N	N
	D2	N	N	N	A	C	C	N

Figure 49 - Successful Outgoing Call

1. At time t1 both connections are in the Null state. Typically, a voice terminal would be on hook.
2. At time t2 the call initiator will be off hook and possibly dialling digits. This state is entered by receiving a Service_Initiated event. The event says that device D1 has changed to the Initiate state.
3. If device D1 aborts the call then the state returns to t1 and is indicated via the Call_Cleared event. The event says that all connections in call C1 have returned to Null.
4. At time t3 the call initiator will have finished dialling and will be receiving feedback from the CSTA telecommunications network. The event Call_Originated indicates that the device D1 moves to the Connect state.
5. The Call_Delivered event indicates that device D2 enters the Alerting state. This is depicted at time t4. At this stage, typically the caller is listening to ring back, and the called party is listening to ringing.
6. When device D2 answers the call, the state shown at time t5 is entered. This is reported by the Call_Established event. It indicates that device D2 entered the Connect state. Typically, two parties would now be talking.

11.4.1.2 Clearing an Established Call

Using the same figure provided in "Successful Outgoing Call", call clearing can progress in one of two ways. If one device drops, for example device D1, then a Device_Dropped event would be received, and the state would match the one shown for the time t6. When the call ends, either from the CSTA network nulling the device D2 connection, or if the call had originally been cleared via the CSTA Clear_Call request, then the connection states return to null, as depicted at time t7. Note that the states at times t1 and t7 are identical.

11.4.1.3 Call Failure

Call C1 at time:

Devices		t1	t2	t3	t4	t5
	D1	N	I	C	F	N
	D2	N	N	N	N	N

Figure 50 - Call Failure at Device D1

The Call__Failed event can come during call establishment. Depending on different types of diversion, the call can fail until the point that a Call__Established event is received. If the call had failed, the initiating connection enters the Failed state and the caller typically listens to a busy or reorder tone. After the caller hangs up, device D1 returns to Null. An example of a call that fails is shown above. In this example, the call progresses to time t3. It then encounters Call__Failed which indicates that device D1 failed to create the call, and that the state shown at time t4 is in effect. Similarly, as shown below, Call__Failed may have been the event, which might indicate that device D2 is busy. These states differ, and allow an application to decide whether it can use a call completion feature like Camp-on or Intrude.

Call C1 at time:

		t1	t2	t3	t4	t5
Devices	D1	N	I	C	C	N
	D2	N	N	N	F	N

Figure 51 - Call Failure at Device D2

11.4.1.4 Call Abandonment After No Answer

Call C1 at time:

		t1	t2	t3	t4	t5	t6
Devices	D1	N	I	C	C	N	N
	D2	N	N	N	A	A	N

Figure 52 - Call Abandonment after no Answer

The above matrix shows a normal call progression up until time t4. At this point a Device__Dropped event is reported, indicating that device D1 has returned to the Null state. Soon after this the CSTA telecommunications domain will set the remaining device D2 to Null as well and report a Call__Cleared event. This is depicted in the state at time t6.

11.4.1.5 Network__Reached

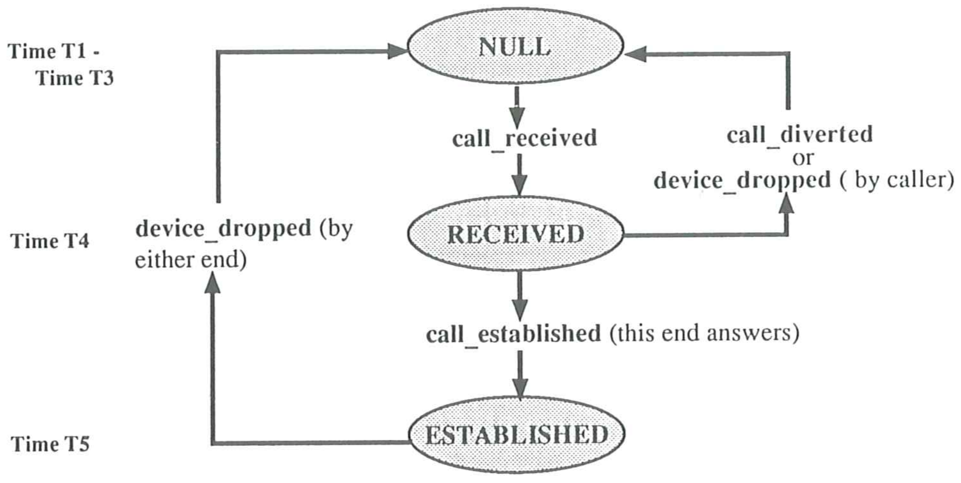
Call C1 at time:

		t1	t2	t3	t4	t5
Devices	D1	N	I	C	C	N
	D2	N	N	N	C	N

Figure 53 - Network__Reached

In the above matrix, the state at time t4 was entered because of a Network__Reached event report. This event report indicates to the application the possibility that no event reports will be provided for connections outside the CSTA telecommunications domain. The application should assume (as is shown in the figure) that the connections in the other network are in the connect state until a Call__Cleared event is received (as shown at time t5).

11.4.2 Incoming Call


Time T1 -
Time T3
NULL
call_received
call_diverted
or
device_dropped (by caller)
Time T4
device_dropped (by either end)
RECEIVED
call_established (this end answers)
Time T5
ESTABLISHED

Note: The annotated times relate the diagram to the following descriptions

Figure 54 - Incoming Call

The events and states reported for an incoming call are exactly those shown and reported for an outgoing call. After all, they are but one call. Differences in the operation of CSTA applications may come from the time at which applications start receiving the event reports. For example, in the "Successful Outgoing Call" scenario, an application that uses a monitor defined to give events in calls for all calls associated with a device would receive different events if it monitored calls at device D1 or calls at device D2. The figure was shown for the events visible to a monitor for calls at device D1 (and, of course, for any monitor for Call C1). If the application had been monitoring calls at device D2, the figure would have started at time t4 and progressed identically from that point onward.

This demonstrates a more general problem with monitoring - that of applications starting to monitor the call at some time after the call has started. These applications will start receiving event reports that indicate the current state of different endpoint within that call. Building the call state image will require an event from each connection participating in the call. This may take time, so the Snapshot service allows an application to build that state image at once. At any time, an application can ask for the snapshot of a call and be provided the state information for that call at that time.

11.4.3 Holding Calls

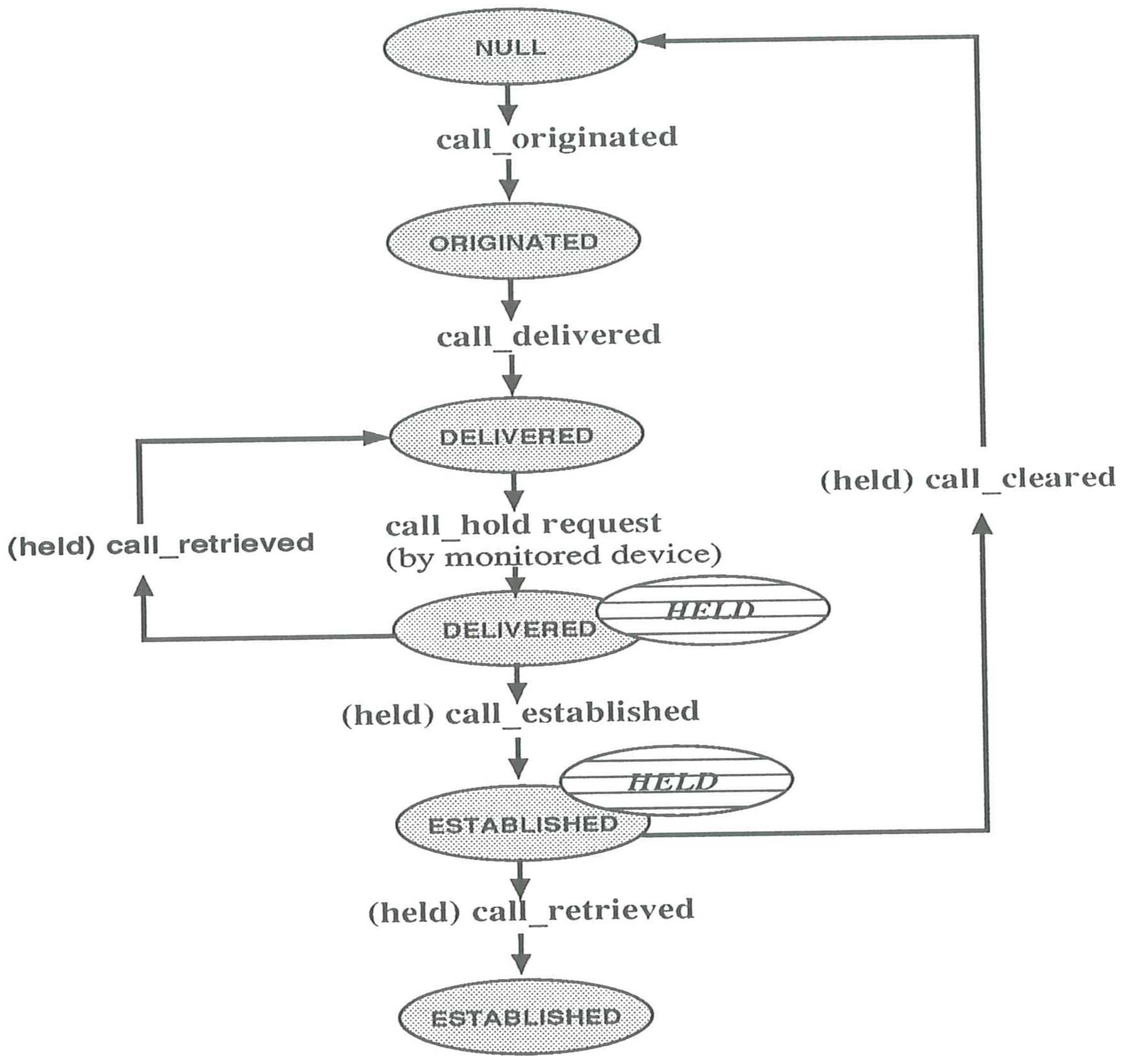


Figure 55 - Holding Calls

These scenarios introduce the notion of an auxiliary state to represent the held condition of the call. This idea has been adopted from the "dimensioned state space" described in CCITT Rec. Q.932. The Held state is a pervasive state and the precise details (such as when a call can be placed on hold) can vary between switches. To demonstrate the holding of a call, the scenario shows an outgoing call which reaches the Delivered CSTA state when it is placed on hold. The events and states are for the party performing these operations (either itself or via a CSTA request). Many other scenarios can be considered where the hold request is issued at a different time (in particular, when the call is established). In the following matrix description of the holding scenario, the auxiliary state is contained in the matrix, which is a multi-dimensional state representation.

Call C1 at time:

		t1	t2	t3	t4	t5	t6	t7
Devices	D1	C	H	C	C	H	H	C
	D2	C	C	C	H	H	C	C

Figure 56 - Three Hold Services

The above table shows the interaction of three Hold services on a simple call. At time t1 both parties are connected. The Call__Held event puts the call in the state at time t2. At this time device D1 has device D2 "on hold". Call__Retrieved indicates that device D1 has rejoined the call and the state is at time t3. The remaining states are traversed as follows: Call__Held yields t4; Call__Held yields t5 (both parties are holding); Call__Retrieved yields t6; and finally Call__Retrieved yields t7. At this time both parties are speaking once again.

11.4.4 Multiple Call Support

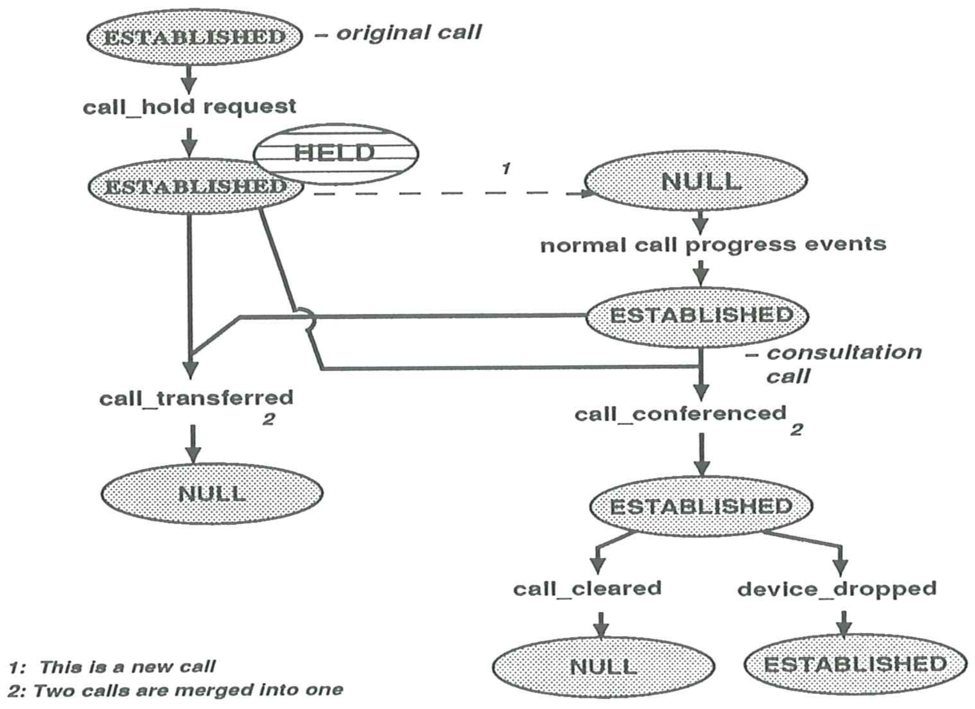


Figure 57 - Multiple Call Support

The figure shows typical CSTA states and events supporting multiple calls. The diagram shows two calls; the established call (on the left) is placed on hold and a new consultation call is initiated (shown on the right). This operation may have been initiated manually or via CSTA. The new consultation call will proceed in the same way as any other outgoing call and is not detailed on the diagram. The first scenario considers a supervised transfer where the consultation call is established before the transfer operation is used. Other types of transfer (for example blind transfer) can be achieved by requesting the transfer prior to the establishment of the consultation call. In this case the Call__Transferred event occurs in other CSTA states such as Delivered. In either case, transfer joins the two calls with the initiating device dropping out. The second scenario shows a conference request which results in a Call__Conferenced event.

The following set of matrices show relationships between multiple devices and multiple calls. Unlike previous columns, these columns show calls. The states at different times are shown as entire matrices labelled with their time. Finally, this example assumes an established call between device D1 and D2 at time t1.

Calls at time t1

		C1	C2
Devices	D1	C	N
	D2	C	N
	D3	N	N

Calls at time t2

		C1	C2
Devices	D1	C	N
	D2	C	N
	D3	N	C

Calls at time t3

		C1	C2
Devices	D1	C	A
	D2	C	N
	D3	N	C

Figure 58 - Device D1 receives a 2nd Call, C2

In the above matrices, a call exists between devices D1 and D2 at time t1. The event report Call__Originated moves the state diagram to time t2. The next event report, Call__Delivered, shows the transition to the state diagram for time t3. Note Call__Delivered indicates that the device D1 changes to the Alerting state. At this point D1 and D2 are still talking, but D3 is ringing D1 as well.

Calls at time t4

		C1	C2
Devices	D1	H	A
	D2	C	N
	D3	N	C

Calls at time t5

		C1	C2
Devices	D1	H	C
	D2	C	N
	D3	N	C

Calls at time t6

		C1	C2
Devices	D1	H	N
	D2	C	N
	D3	N	C

Figure 59 - Device D1 answers Call C2, then Drops it

In the set of matrices above, the next three events are depicted. First, D1 puts Call C1 (with D2) on hold which yields a Call__Held event report and moves to time t4. D1 then answers the ringing call from D3, which results in a Call__Established event report and moves to time t5. At this point D1 and D3 are talking and D1 has D2 on hold. Three examples follow on from this point:

1. D1 and D3 finish their conversation, D1 hangs up as shown at time t6 and indicted by a Device__Dropped event report. Next Call__Cleared is reported and probably Call__Retrieved is also reported (state matrices are not shown for these two events).

2. Once again, starting with state t5 as shown above it is also possible to establish a conference. This is shown below.

Calls at time t6

		C1	C2	C3
Devices	D1	N	N	C
	D2	N	N	C
	D3	N	N	C

Calls at time t7

		C1	C2	C3
Devices	D1	N	N	N
	D2	N	N	N
	D3	N	N	N

Figure 60 - D1 conferences, then clears the Call

D1 conferences the calls with D2 and D3. This is indicated by Call__Conferenced event which moves to state t6 shown above. At some later time t7 in this example, D1 clears the entire call (as opposed to dropping off) and Call__Cleared event is reported.

3. Finally, starting with state t5 as shown above, it may also be possible to transfer the call. This is shown below.

Calls at time t6

		C1	C2	C3
Devices	D1	N	N	N
	D2	N	N	C
	D3	N	N	C

Calls at time t7

		C1	C2	C3
Devices	D1	N	N	N
	D2	N	N	N
	D3	N	N	N

Figure 61 - D1 transfers the Call, then the Call clears

D1 transfers the call from D2 to D3. This is indicated by a Call__Transferred event which moves to state t6 shown above. At some later time t7 in this example, D2 clears the call and a Call__Cleared event is reported.

11.4.5 Tracking a Call

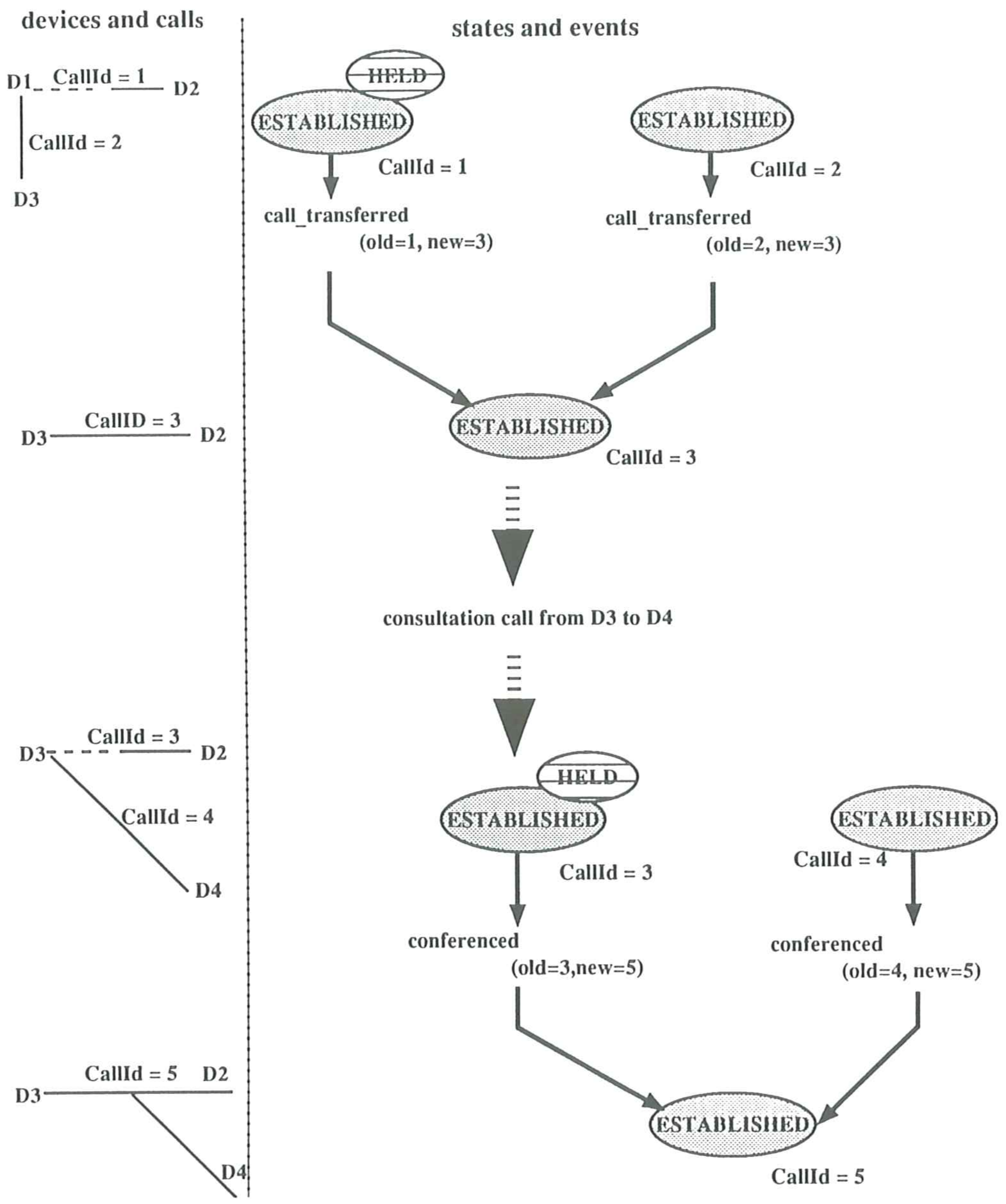


Figure 62 - Tracking a Call

The previous figure demonstrates the tracking characteristic of a call monitor applied at device D1. On the left are the CSTA Devices and Calls; on the right are device abstracted CSTA states and events.

Tracking a call is an important ability that CSTA applications need to have. The following scenario demonstrates a method that can be used to track calls within a CSTA switching sub-domain. Assume that a call is established between D1 and D2, and that D1 is holding that call. Assume also that a second call is established, this one from D1 to D3. This state is that at time t5 in the previous text, and is repeated as the initial state, t1, below.

Calls at time t1

		C1	C2	C3
Devices	D1	H	C	N
	D2	C	N	N
	D3	N	C	N

Calls at time t2

		C1	C2	C3
Devices	D2	N	N	C
	D3	N	N	C

Figure 63 - D1 transfers the Call with D2 to D3

D1 transfers the call from D2 to D3. This is indicated by a Call_Transferred event which moves to state t2 shown above. At some later time, t3 in this example, D2 executes a Direct, Unsupervised Transfer of the call to D4. This results in a Call_Transferred event report. Finally, D4 answers the call, as shown at t4, a Call_Established is received and D3 and D4 are talking.

Calls at time t3

		C3	C4
Devices	D2	N	N
	D3	N	C
	D4	N	A

Calls at time t4

		C4
Devices	D3	C
	D4	C

Figure 64 - D2 transfers the Call to D4

The previous example shows how a call, originally between D1 and D2 is tracked to a call between D2 and D3 and then to a call between D3 and D4. In this final permutation, none of the original participants are left on the call, but the context or voice path has been followed. This tracking would continue until the Call_Cleared (e.g. a Call_Cleared event was reported), or until the call left the CSTA sub-domain (e.g. Network_Reached followed by Device_Dropped).

11.4.6 Events Prompted by the Remote Device

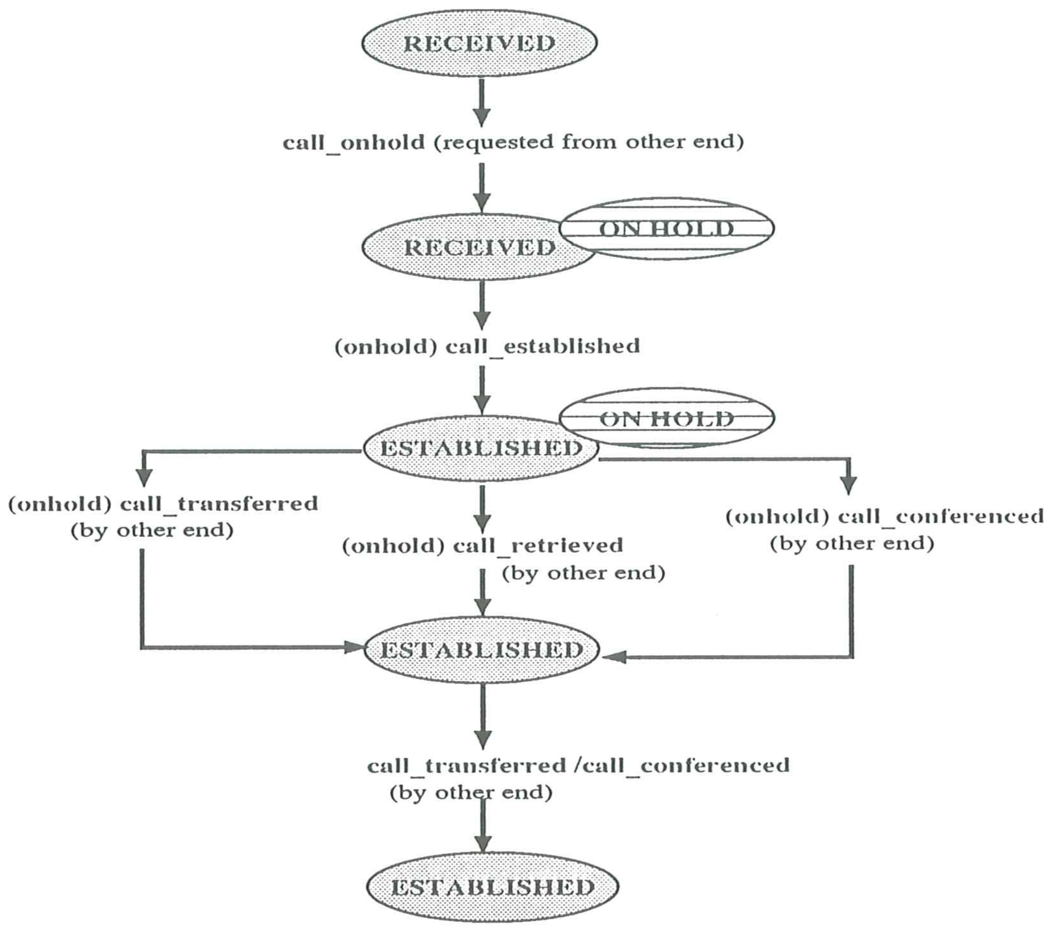

RECEIVED
call_onhold (requested from other end)
RECEIVED
ON HOLD
(onhold) call_established
ESTABLISHED
ON HOLD
(onhold) call_transferred
(by other end)
(onhold) call_retrieved
(by other end)
(onhold) call_conferenced
(by other end)
ESTABLISHED
call_transferred /call_conferenced
(by other end)
ESTABLISHED

Figure 65 - Events Prompted by the Remote Device

Events can be prompted by the other end of the call as shown in the figure. This shows an incoming call to a monitored device which is placed on hold by the caller. Various scenarios are shown to reflect events prompted by actions at the calling device (at the other end of the call).

These scenarios show events and states reported (for a monitored device) due to Switching Functions initiated at the remote end of the call. The initial scenario is that an incoming call arrives at the monitored device and is subsequently answered. The figure shows events reporting various changes in CSTA state that have been initiated at a far end of the call (as opposed to requested by this monitored device).

11.4.7 Queueing (inbound)

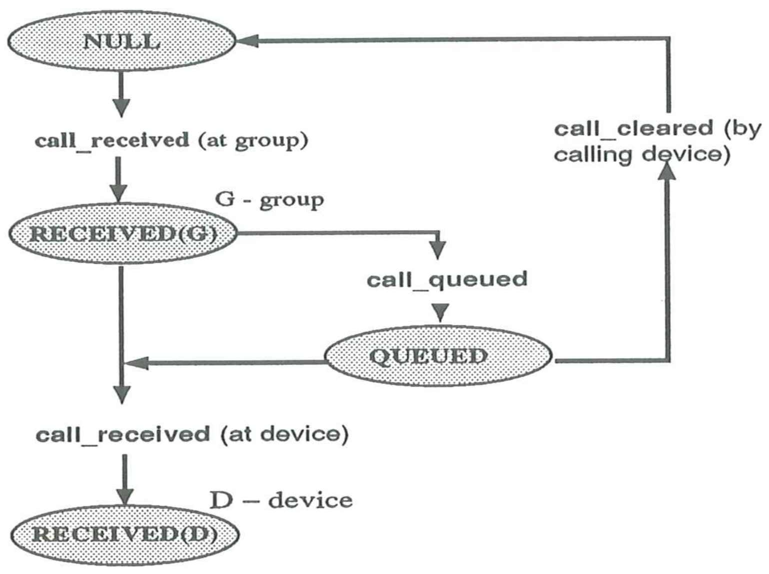


Figure 66 - Queueing (inbound)

The figure shows a typical scenario involving an incoming call arriving at a group (which assumes that monitoring has been initiated for the group). Three typical cases are shown:

1. The call arrives at a group and is distributed to a particular device.
2. The call is queued, typically because no agents are available, before distribution to a particular device.
3. The caller abandoning the call when (queued) in the group.

11.4.8 Predictive Dialling

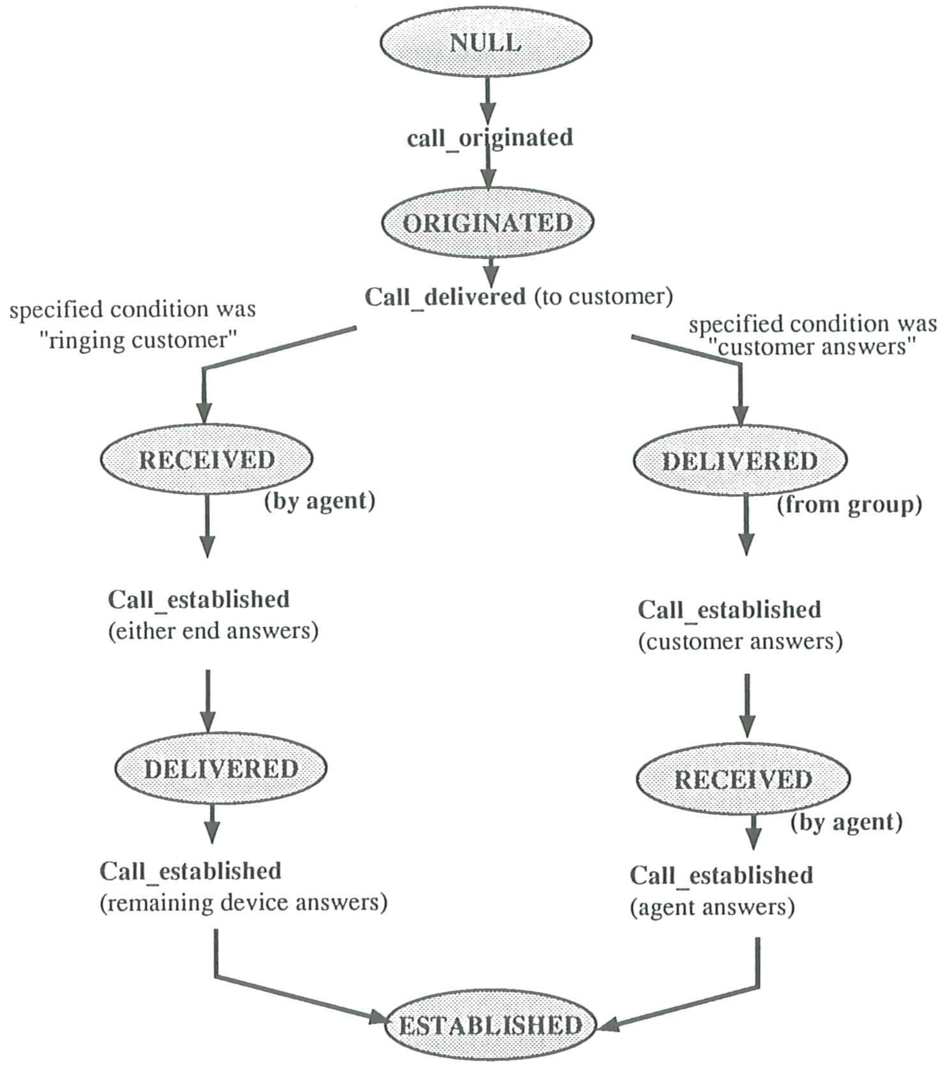

NULL
call_originated
ORIGINATED
Call_delivered (to customer)
specified condition was "ringing customer"
specified condition was "customer answers"
RECEIVED
(by agent)
DELIVERED
(from group)
Call_established (either end answers)
Call_established (customer answers)
DELIVERED
RECEIVED
(by agent)
Call_established (remaining device answers)
Call_established (agent answers)
ESTABLISHED

Figure 67 - Predictive Dialling

11.5 CSTA Events

Event report messages will be sent from server to client. For the Telecommunications domain service, those events are sent from the telecommunications network to the computing network. Each event is a message that indicates a change in state of one or more connections in the CSTA network. Generally, the event report messages will contain call__id and device__id to indicate the connection that has changed state.

The messages described can apply to:

1. A single connection.
2. Multiple connections within a call.
3. Multiple connections within multiple calls.

The following list describes the events and the information they provide. Each event relates to the example connection state matrices in the following way:

- The event specifies the resultant state has been achieved regardless of any previous state.

- The initial state(s) is given purely for example to put the event into normal telecommunications context.
- Each event report is a name (macro) for the new set of connection states.

1. Call__Delivered - this event is sent when the first "alerting" (tone, ring, etc.) is applied to a device or when the server detects that "alerting" has been applied to a device.

   Server features might allow multiple devices to be alerted at the same time. When this happens, a follow on Call__Established event for the device might have a different connected device__id than the called device__id passed in the previous Call__Delivered event for that device. In addition, every time a Call__Delivered event for a device is followed by another Call__Delivered event for another device, it implies that the first device is no longer involved in the call and that the call has been redirected.

Call C1 at times:

		t1	t2
Device	D1	N	A

Figure 68 - Call__Delivered

The following items are included with this event:

- call__id: the call identifier;
- dynamic device__id: the dynamic device identifier for this endpoint in this call;
- static device__id: the identifier of the alerting device. This value may be null (0). See clause 9 for device types;
- calling static device__id: the identifier of the calling device. This value may be null (0);
- called static device__id: the identifier of the originally called device. This value may be null (0).

NOTE 7
It has been suggested that a monitor service be defined so that the Call__Delivered event report be sent only to applications monitoring the device which initiates the call. In this service Call__Received would be sent to applications monitoring the called device. What applications that monitor other devices in the call or the call itself receive is for further study in this proposal.

2. Call__Conferenced - this event is sent when a device has been conferenced into an existing call.

   This event applies to multiple connections within multiple calls.

Calls at time t1

Devices		C1	C2	C3
	D1	H	C	N
	D2	C	N	N
	D3	N	C	N

Calls at time t2

Devices		C1	C2	C3
	D1	N	N	C
	D2	N	N	C
	D3	N	N	C

Figure 69 - Call__Conferenced

The following items are included with this event:

- call__id: one old (pre-conference) call identifier;
- call__id: the other old (pre-conference) call identifier;
- new__call__id: the new (post-conference) call identifier.

And for each device in the conference:

- dynamic device__id: the dynamic device identifier for the endpoint;
- static device__id: the identifier of the device. This may be null (0).

3. Call__Cleared - this event is sent when a call is torn down. Normally this occurs when the last remaining device disconnects from the call. It can also occur when a call is immediately dissolved as a conference call can be by the conference controller.

Call C1 at times:

Devices		t1	t2
	D1	C	N
	D2	C	N
	D3	C	N

Figure 70 - Call__Cleared

This event applies to all connections within a call.

The following items are included with this event:

- call__id: the call identifier.

4. Service__Initiated - this event is sent when telephony service is initiated at a monitored device. The server issues this event at providing "dial-tone". Note: this event will not be sent from functional terminals and will not be present for calls that are set up without receiving dial-tone, like CSTA requested calls.

Call C1 at times:

		t1	t2
Device	D1	N	I

Figure 71 - Service__Initiated

This event applies to a single connection.

The following items are included with this event:

- call__id: a "service" identifier to distinguish this transaction from other calls that may be present at the device. This service will typically become a call, and this identifier will later be used to represent that call;
- static device__id: the identifier of the device;
- dynamic device__id: the handle of the device.

5. Call__Originated - this event is sent when a call is initiated from a monitored device. The server issues this event at call origination.

Call C1 at times:

		t1	t2
Device	D1	I	C

Figure 72 - Call__Originated

This event applies to a single connection.

The following items are included with this event:

- call__id: the call identifier;
- static device__id: the identifier of the called device;
- calling__static device__id: the identifier of the calling device.

6. Call__Transferred - this event is sent when an existing call is transferred to another device and the device requesting the transfer has been dropped from the call. The transferring device does not appear in any future feedback for the call.

Calls at time t1

Devices		C1	C2	C3
	D1	H	C	N
	D2	C	N	N
	D3	N	C	N

Calls at time t2

Devices		C1	C2	C3
	D1	N	N	N
	D2	N	N	C
	D3	N	N	C

Figure 73 - Call__Transferred

This event applies to multiple connections within multiple calls.

The following items are included with this event:

- call__id: one old (pre-transfer) call identifier;
- call__id: the other old (pre-transfer) call identifier;
- new__call__id: the new (post-transfer) Call Identifier.

And for each device in the resulting call:

- dynamic device__id: the dynamic device identifier for the endpoint;
- static device__id: the identifier of the device. This may be null (0).

7. Call__Established - this event is sent when the server detects that a device answers a call.

Call C1 at times:

Device		t1	t2
	D1	A/I	C

Figure 74 - Call__Established

This applies to a single connection.

The following items are included with this event:

- call__id: the call identifier;
- dynamic device__id: the dynamic device identifier for the endpoint;
- static device__id: the identifier of the connected device. This parameter may be null (0).

8. Network__Reached - this event is sent when a call is cut through the CSTA telephony network boundary to another network (sent to an outgoing trunk). This event implies that there may be no additional device feedback, except disconnect/drop, provided for this end of the call. A Network__Reached event is never sent for devices connected directly to the CSTA network.

Call C1 at times:

		t1	t2
Device	D1	N	C

Figure 75 - Network__Reached

This event applies to a single connection.

The following items are included with this event:

- call__id: the call identifier;
- dynamic device__id: the dynamic device identifier for this endpoint in this call;
- static device__id: the identifier of the requested or last redirected number.

9. Device__Dropped - this event is sent when a device in a call disconnects or is dropped from the call - except for a call transfer operation for which this event does not apply for the party doing the transfer.

Call C1 at times:

		t1	t2
Device	D1	C	N

Figure 76 - Device__Dropped

This event applies to a single connection.

The following items are included with this event:

- call__id: the call identifier;
- dynamic device__id: the dynamic device identifier for the dropped device;
- static device__id: the identifier of the disconnecting or dropped device. This parameter may be null (0).

10. Call__Held - this event is sent when the server detects that an existing call has been held by one of the devices on the call.

Call C1 at times:

		t1	t2
Device	D1	C	H

Figure 77 - Call__Held

This event applies to a single connection.

The following items are included with this event:

- call__id: the call identifier;
- dynamic device__id: the dynamic device identifier for the device which activated hold;
- static device__id: the identifier of the device which activated hold.

11. Call__Queued - this event is sent when a call queues. Queueing may occur at an ACD, hunt group, or other device; or queueing may occur during network routing without an associated device.

Call C1 at times:

		t1	t2
Device	D1	?	Q

Figure 78 - Call__Queued

This event applies to a single connection. This description refers to queues that are logically part of a call routing system and do not participate in the call. Other queues actually answer and participate in the call and would be modelled as a device rather than a connection state. Further study is needed on queues.

The following items are included with this event:

- call__id: the call identifier;
- dynamic device__id: the dynamic device identifier for the device for the queue if it participates in a call. This parameter may be null (0);
- number__of__calls__in__queue;
- static device__id: the pilot static device__id of the queue. This parameter may be null (0).

12. Call__Retrieved - this event is sent when the server detects that a previously held call at a directly connected device has been reconnected.

Call C1 at times:

		t1	t2
Device	D1	H	C

Figure 79 - Call__Retrieved

This event applies to a single connection.

The following items may be included with this event:

- call__id: the call identifier;
- dynamic device__id: the dynamic device identifier for the reconnecting device;
- static device__id: the identifier of the reconnecting device.

13. Call__Failed - this event is sent when a call cannot be completed because all the trunks are busy, the calling device is not permitted service, required resources are not available, no agents logged-in or available in an ACD, a queue is full, or a reorder tone is detected by the server from another network.

Call C1 at times:

		t1	t2
Device	D1	?	F

Figure 80 - Call__Failed

This event applies to a single connection.

The following items are included with this event:

- call__id: the call identifier;
- device__id: the identifier of the busy device (when applicable);
- cause: indicates the cause of the reorder, such as:
  - . trunks busy
  - . device busy
  - . no available agents
  - . queues full
  - . resources not available (i.e. no time slot, no call records)
  - . call not answered
  - . reorder tone

SECTION IV - CSTA APPLICATIONS

12. CSTA SERVICE BOUNDARY

12.1 Introduction

With reference to the material of clause 8.1.8 dealing with the structure of the application layer, this clause examines the decomposition of certain CSTA Services into more primitive services. This will lead to an examination of the position of the CSTA Service Boundary and the services applicable at that boundary. The material of this clause will lead, at a later date, to Stage II Service Descriptions.

12.2 Service Decomposition Functional Elements

Each of the Switching System Services defined in the Service Description clause (see clause 10.2) is specified in accordance with Stage I of the CCITT Three-Stage service description method (see CCITT Rec. I.130 and Rec. Q.65). This means that each is expressed in terms of:

1. The appearance of the service to the computer application.
2. The behaviour, in terms of telecommunications activities, at the Switching Function peripherals.

Some services, however, may be presented to the switching application in more than one way. Consider, for example, the Alternate_Call service. This might equally well be performed by requesting the Hold_Call service (on the active call) followed by (or simultaneously with) the Retrieve_Call service (on the held call). It is possible to perform a similar mapping process for other services in clause 10.2.

This mapping may be modelled by identifying one or more Service Decomposition Functional Elements (SDFE) for each service whose function is to map the 'higher level' service to the 'lower level' service(s). The SDFE may be null in the case of sufficiently primitive services, such as Hold_Call.

Consider again the example of the Alternate_Call. The mapping already described may be performed by the Alternate_Call SDFE, as in the figure below.

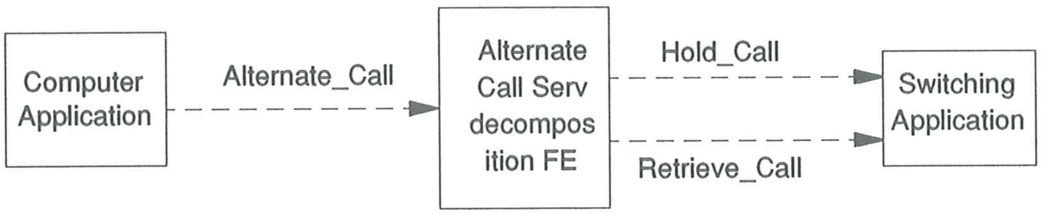


Figure 81 - Service Decomposition Example

12.3 Location of Decomposition Functional Elements

It is possible to locate these Service Decomposition Functional Elements at the most 'appropriate' location.

NOTE 8
There are no guidelines yet on what constitutes an appropriate location. It may be the most convenient, economical, elegant, efficient, etc. This will be decided later on in the standardization process.

Consider again the example of the Alternate_Call service. If the SDFE is located in/with the Computer Application, then the CSTA Application Service Boundary (CASB) requires only the Hold_Call and Retrieve_Call service primitives in order to provide the Alternate_Call service, as illustrated in the figure below.

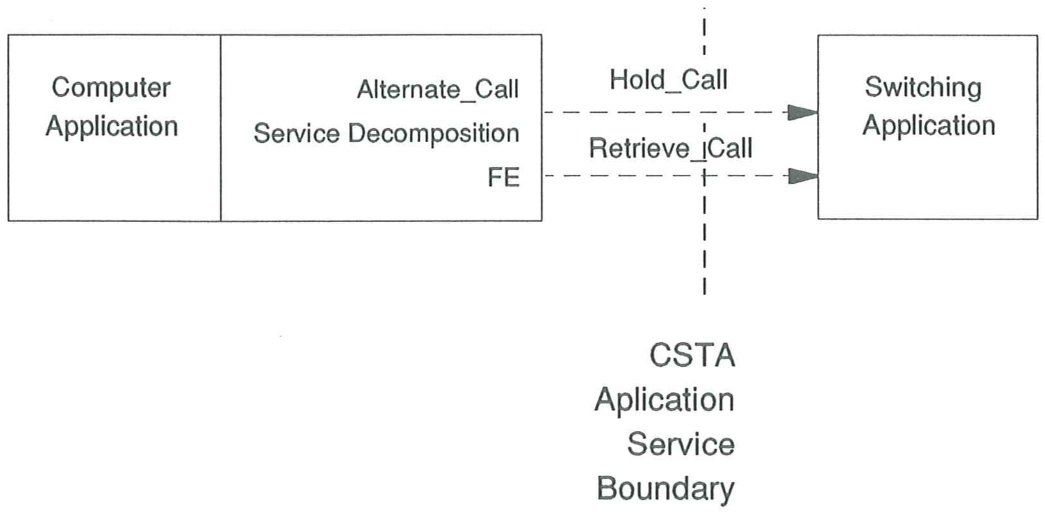

Computer Application
Alternate_Call Service Decomposition FE
Hold_Call
Retrieve_Call
Switching Application
CSTA Aplication Service Boundary

Figure 82 - Location of SDFE in Computer Function

NOTE 9
This situation is sometimes referred to as saying that the "intelligence" for the service resides in the Computing Function.

Alternatively, if the SDFE is located in/with the switching application, then the CASB requires the Alternate__Call service primitive in order to provide this function, as illustrated in the figure below.

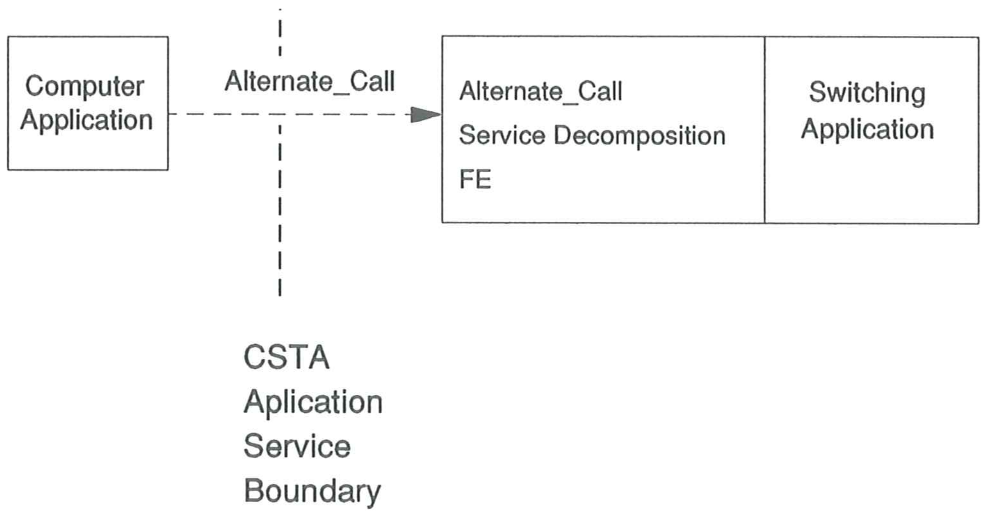

Computer Application
Alternate_Call
Alternate_Call Service Decomposition FE
Switching Application
CSTA Aplication Service Boundary

Figure 83 - Location of the SDFE in Switching Function

NOTE 10
This situation is sometimes referred to as saying that the "intelligence" for the service resides in the Switching Function.

A similar functional decomposition will be performed for each service in clause 10.2, in order to identify the necessary interfaces and the corresponding set of service primitives. This information will provide the Stage II Service Descriptions.

13. APPLICATION LAYER STRUCTURE

13.1 Introduction

The Functional Architecture, contained in clause 8, described how CSTA is the functional integration of a computing system and a switching system. It identified a client/server model as appropriate for describing the communications aspect of this integration and positioned this mechanism in the OSI application layer. This clause provides some brief tutorial information on OSI Application Layer Concepts followed by identifying the CSTA application layer service definition and corresponding application protocol to be standardized:

1. The service is a local interface provided to the CSTA processing component so that it can communicate with its peer CSTA processing component (in order to support the CSTA application).

2. The application protocol (i.e. a set of semantic and syntactic rules for communication) will be compatible with the Application Layer Structure concepts described in the OSI Reference Model.

The mapping of the CSTA Application layer protocol elements and messages onto the lower layer services is outlined in the following clause on Interconnection Services Architecture.

13.2 OSI Reference Model

The Basic Reference Model for Open Systems Interconnection (OSI) defined in ISO 7498 provides a common basis for the co-ordination of standards concerned with systems interconnection.

The OSI Reference Model serves as a framework for the definition of services and protocols of CSTA.

This clause is concerned primarily with the application layer of the OSI reference model. The Application layer structure is described comprehensively in ISO 9545 but a brief summary of the key concepts is provided in this subclause.

A user-application example showing a possible relationship between the OSI Reference Model and the CSTA architecture is given in Appendix B.

NOTE 11
The figure at the end of sub-clause 13.2 relates the following OSI concepts to CSTA.

13.2.1 General

OSI standards support the communication requirements of applications (i.e. information processing tasks) requiring coordinated processing in two or more real open systems. Examples of such applications are CSTA applications as introduced in the Functional Architecture clause. The Application Layer defines procedures for the support of distributed information processing and is supported by the lower layers of OSI.

13.2.2 Application Process

The cooperative operation of real open systems is modelled in terms of interactions between application-processes (APs) which process information sent to and/or received from APs in other real open systems.

An AP is a static representation of a set of resources, including processing and communication resources, within a real open system. The activity of an AP is dynamically represented by the AP-invocation (see figure at end of 13.2).

When an AP wishes to communicate with another AP it will invoke an instance of an Application Entity (AE) in the application layer of its own open system.

13.2.3 Application Entity

An AE is a static representation of a set of communications capabilities of an AP. The activity (instance) of an AE is dynamically represented by the AE-invocation. A particular AE-invocation may not necessarily use the full communications capabilities of the AE.

An AE can encompass one or more application associations (see 13.2.5). Each association defines one application context (see 13.2.6.) AE-invocations communicate using Application Service Elements (ASE). This structure is shown below.

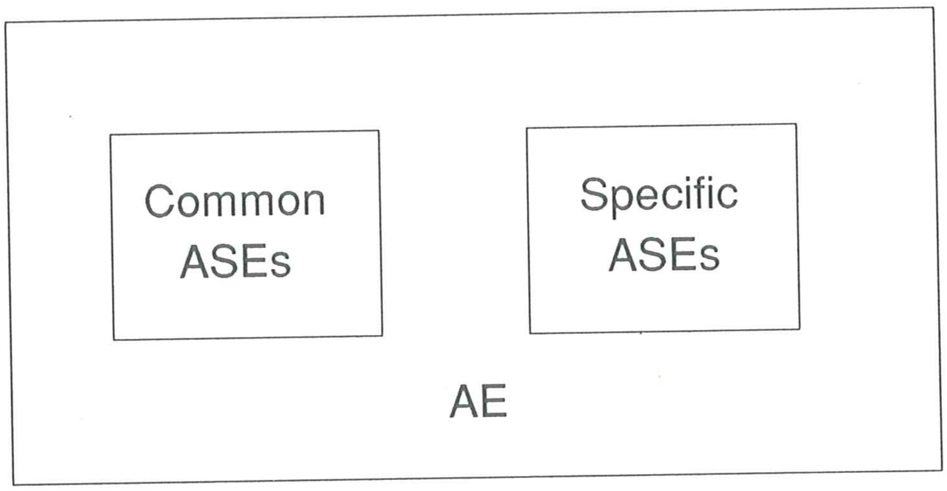


Figure 84 - Application Entity

13.2.4 Application Service Elements

An Application Service Element (ASE) is a set of functions that provide OSI communication capabilities for the interworking of AE-invocations for a specific purpose. The communication capabilities of an ASE are defined by the specification of a set of application-protocol-data-units (APDU) and the procedure governing their use. An AE-invocation may be composed of one or more ASEs.

There are two categories of ASEs:

Common ASEs which provide general capabilities useful to a variety of applications. Examples of these common ASEs are:

1. ROSE (Remote Operations Service Element), and
2. ACSE (Association Control Service Element)

Specific ASEs which provide a particular capability required by specific applications such as the CSTA applications.

An ASE identifies one or more operations and specifies how those operations are used; that is, it specifies which peer AE may invoke which operations and in what order. An ASE is a set of operations.

CCITT Rec. X.219 (ROSE) defines Application-Service-Element macros which can be used to specify an ASE formally. It identifies which operations are contained in the ASE.

13.2.5 Application Associations

An application-association is a cooperative relationship between two AE-invocations for the purpose of communication and coordination of their joint operation. For the case where an AE-invocation involves a number of application associations a Multiple Association Control Function is required to manage the associations (See figure at end of 13.2). An association between AEs may be established implicitly as well as explicitly i.e. previous agreements allow the AE-invocations to communicate without explicit association control. The termination of an application-association results from the action of the related AE-invocations.

An application association is modelled as a Single Association Object which consists of a set of ASEs one of which may be the ACSE (see figure at the end of 13.2).

13.2.6 Application Context

An application-context relates to an application-association. It is the common set of rules for the related AE-invocations to effectively exchange information. An application-association has only one application context. The appropriate application-context is determined during establishment of the application-association.

The definition of an application-context may be written in natural or formal language.

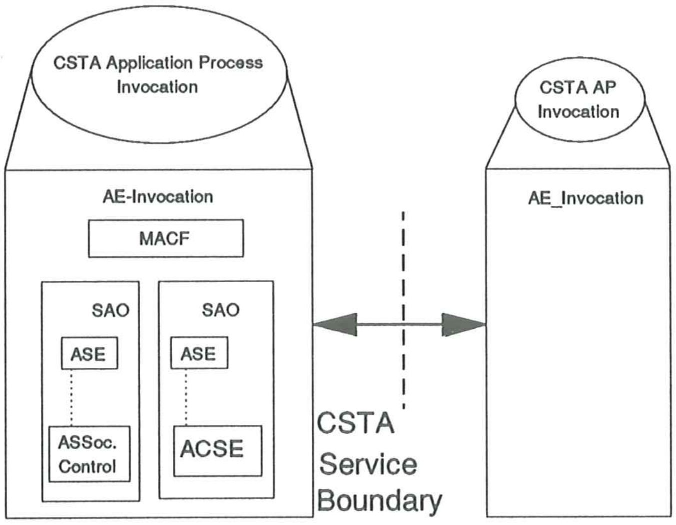


Figure 85 - Application Entity Structure

This diagram makes no assumption about the mapping of CSTA services into OSI Application layer structures beyond identifying the position of the CSTA service boundary as defined in Clause 8. The diagram shows an AP-invocation in the Switching environment communicating with an AP-invocation in the Computing environment.

13.3 CSTA Application Layer Structure

The modelling of CSTA applications in terms of a set of AEs and ASEs, including association and context, follows the OSI Application Layer structure in 13.2.

The selection and maintenance of the route through the CSTA communication network over which CSTA is operating are the responsibility of lower layers of the interface architecture (as described in clause 14) and are not for consideration here.

13.3.1 CSTA Application Process

A CSTA application will be formed from an instance of AP-invocations resident in real systems located within the Computing Function and within the Switching Function.

13.3.2 CSTA Application Entities

When a CSTA Application Process wishes to communicate with another CSTA Application Process, it will invoke an instance of a CSTA Application Entity in the Application Layer of its own open system. A particular CSTA AP may be represented by several AEs.

An AE is composed of one or more ASEs. The composition of a particular AE will depend upon the specific requirements of the particular CSTA application. There will be many possible AEs.

13.3.3 CSTA Association and Context

For a given CSTA Application one or more associations will be established between one instance of a Computing Function application entity and one instance of a Switching Function application entity (i.e. between two AE-invocations). Association between CSTA AEs could be provided in several ways including by the ACSE (see 13.2.4).

Each association has a context which specifies the set of ASEs involved and the rules by which the associated AE-invocations exchange information. As noted in 13.3.2 there will be many possible CSTA applications and therefore many possible application contexts.

No assumption is made at this stage about the lifetime of an association, which may be an indefinite length of time, the duration of one call or even the duration of one message transaction.

13.3.4 CSTA Application Service Elements

A CSTA AE will consist of ASEs and may include common ASEs providing general capabilities and ASEs specific to the CSTA communication requirements.

CSTA specific ASEs are the communications elements which support the CSTA Services of Clause 10. There are various possible mappings between CSTA services and CSTA-ASEs.

Examples of common ASEs are described in sub-clause 13.4

13.3.5 CSTA Specification

CSTA is the standardization of the communications between the Switching Function and the Computing Function at the CSTA Service Boundary as shown in figure at end of 13.2 (and in Clause 8). This will include:

1. Identification of common ASEs.
2. Specification of specific CSTA ASEs.
3. Specification of the CSTA application protocol in terms of semantics and syntax of each APDU together with the procedures for their use.
4. Identification of the requirements placed on the lower layers (an introduction to how these requirements can be satisfied is outlined in the following clause).

13.4 Examples of common ASEs

The CSTA AE defined during standardization will include common ASEs where applicable. Typical examples are:

1. Association Control Service Element (ACSE):

   This manages application-associations and provides an appropriate service interface with primitives of the form:

   - A-ASSOCIATE for establishing an application-association;
   - A-RELEASE for terminating an application-association.

2. Remote Operations Service Element (ROSE):

   This provides the ability to manage the execution of remote operations. It has a defined service interface which includes:

   - RO-INVOKE for invoking a remote operation;
   - RO-RESULT for returning the answer;
   - RO-ERROR for handling errors in the requested operation;
   - RO-REJECT for handling interconnection problems.

   The service interface allows various clauses of operation:

   - synchronous;
   - asynchronous reporting success or failure;
   - asynchronous reporting failure only;
   - asynchronous reporting success only;
   - asynchronous with no reporting;

   and also includes appropriate supporting parameters.

14. INTERCONNECTION SERVICE ARCHITECTURE

14.1 General

The CSTA Architecture is modelled as the co-operation of layer 7 application service elements (ASEs) using the services of the underlying layers 1 to 6. The CSTA services should be, as far as possible, independent from a particular protocol stack. It is with this intent that the CSTA interconnection service interface has been introduced in clause 8.1 as a decoupling abstract boundary.

Hence a variety of protocols (OSI and non OSI) could be retained by the implementers to convey the CSTA service primitives from the computing side to the switching side. This selection will be based on a set of criteria that will be largely influenced by the application requirements.

This hypothetical view, however, is subject to a number of constraints that have been extensively studied and described in other contexts. For example:

CCITT Signalling System No.7 (SS7) provides an intervening function between the network layer service provided by the Signalling Connection Control Part (SCCP) and the service expected by the Transaction Capabilities Application Part (TCAP)

CCITT Rec. Q.940 for network maintenance has considered a possible convergence function to support the carrying of a management protocol on lower layers.

This clause will list some of these constraints and criteria that should allow the selection of protocol stacks.

Given the structure, services and protocols of the application layer (described in clause 13), the remaining task is to select, as examples, some preferred protocol stacks and to map the CSTA service elements on the service primitives of the next lower layer. In this context the following concepts are introduced:

1. The use of convergence functions and the related consequences of 'reduced' stacks,

2. The notion that any real network can exist as a subnetwork (of a global network) and that these different subnetworks may be interconnected via intermediate systems (or interworking units).

14.2 Interconnection architecture modelling concepts

For the purposes of defining an Interconnection Architecture, the application layer is conceived as being supported over a standardized abstract (implementation independent) Interconnection service boundary (see clause 8.1).

This boundary is defined in terms of the abstract Interconnection Service primitives needed to support the transfer of CSTA service primitives between application layer client and server entities, irrespective of the Interconnection service provided by the underlying protocol stack.

14.2.1 Convergence Functions

To reconcile the differences, both in functional and Quality of Service (QoS) terms, between the standardized CSTA Interconnection Service Boundary and the services provided by different, actual underlying protocol stacks, the concept of a stack dependent convergence function is introduced. This convergence function maps locally between the desired and provided interconnection services. In some cases it may be that this function also requires some distributed interaction with the peer convergence function (in another real system) in which case a convergence protocol is required. The figure below illustrates these concepts. An example of the use of a convergence function is described in ECMA TR/45 entitled 'Information Interchange for Remote Maintenance at the Interface between Data Processing Equipment and Private Switching Networks'. In that particular case, the convergence function maps the APDUs generated by the common ASEs, ACSE and ROSE onto the layer 3 service elements specified in Standard ECMA-106 (which defines the D-channel protocol at an S interface).

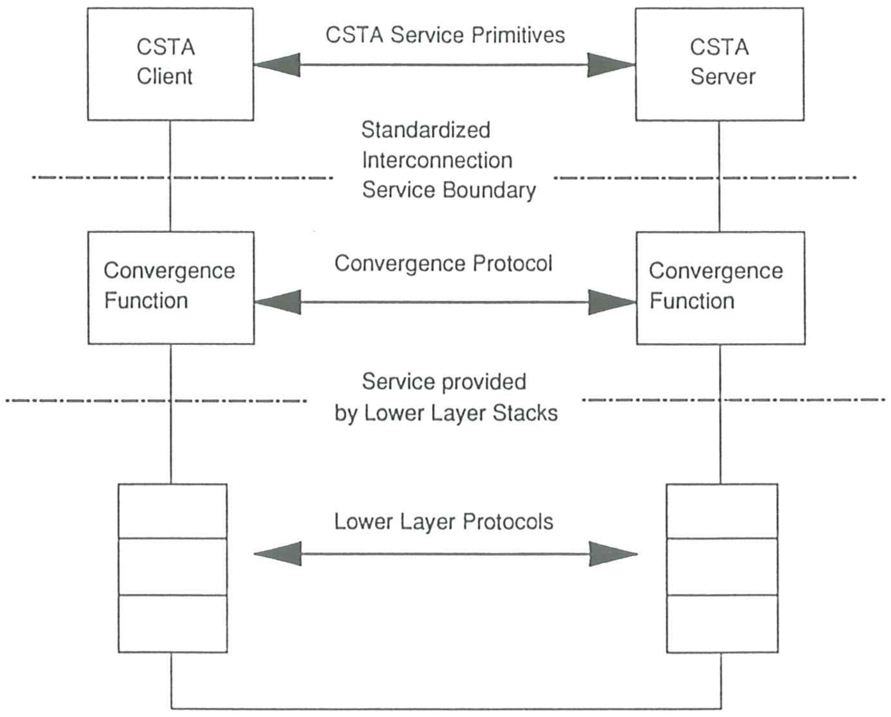


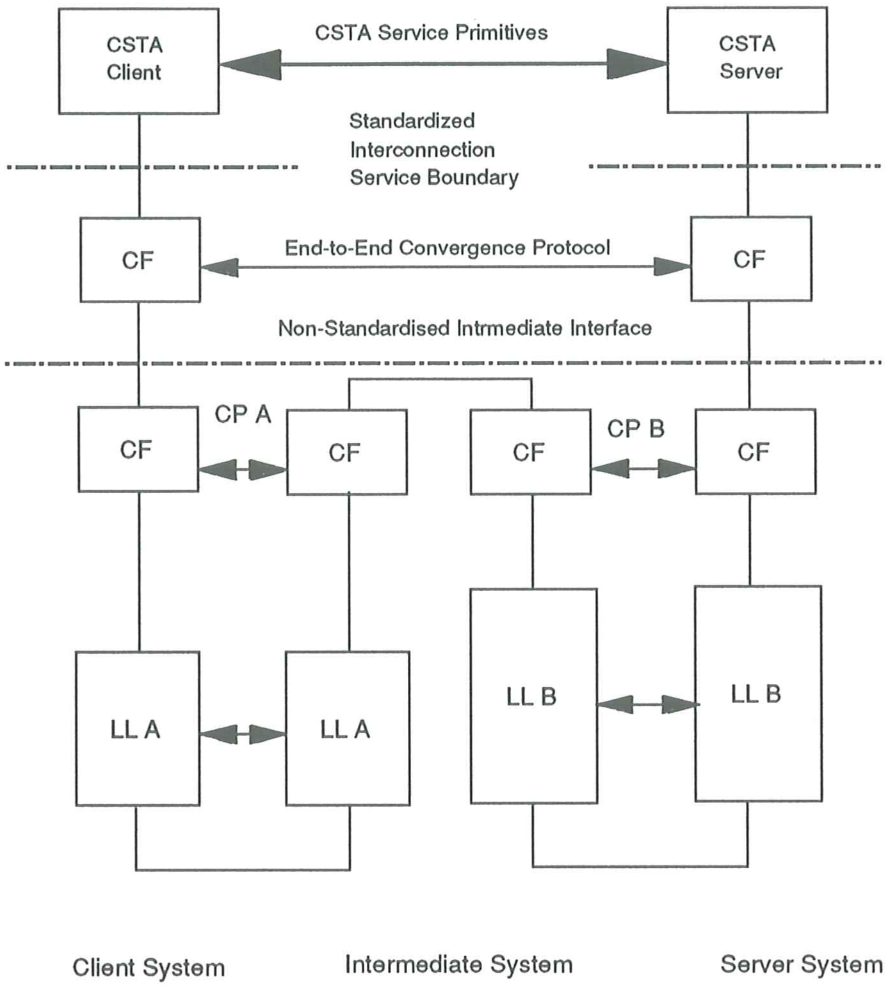
Figure 86 - Concept of Convergence

14.2.2 Subnetworks and their Interconnection

In real networks many different underlying protocols may already be adopted. The OSI Reference Model introduces the concept of 'subnetwork' to accommodate these real networks and this concept is elaborated in ISO 8648 in the Internal Organisation of the Network Layer. These concepts are also developed in ECMA TR/44 ("An Architectural Framework for Private Networks") which describes the use of an Interworking Unit (IWU) as a means of interconnecting a number of subnetworks. The IWU and the subnetworks are all represented as intermediate systems with a certain protocol intervention level (which delineates those protocol layers operated on and those not). Various configurations can be considered for the relative levels of these protocol intervention levels within the IWU and the interconnected subnetworks as described in ECMA TR/44.

The use of an IWU to interconnect subnetworks will normally require consideration of the addressing aspects since addresses are normally local (to the subnetwork) in scope. Other subnetworks will typically interpret non-local addresses and derive information from them to access the appropriate IWU. Component addresses (such as telephone numbers etc) possibly used to actually route messages in one subnetwork will be visible in the other subnetwork as application level objects (eg. a CSTA Static Device Identifier introduced in the Operational Model clause).

A number of options exist with respect to this interconnection mechanism. They are distinguished by the degree of interworking capability they achieve on the basis of full or partial convergence to the standardized CSTA Interconnection Service. As a minimum, convergence must be achieved on the basis of some defined subset of the lower layer stack services and an appropriate convergence protocol. The figure below illustrates these concepts.



CP A, CP B: Convegence Protocols A and B

LL A, LL B: Lower Layer Protocol Stacks A and B

Figure 87 - Concept of Convergence with Incompatible Lower Layer Protocol Stacks

14.3 Criteria for the selection of lower layer protocols

Based on the functional requirements given in the previous clauses, the following criteria can be defined for the selection of the proper lower layer protocols:

1. Speed: CSTA channel throughput, response time
2. Error recovery function, reliability
3. Message size
4. Addressing/routing: transport over switching and/or computing networks
5. Multiplexing of logical links
6. Point to point versus point to multipoint connections
7. Flow control

8. Congestion control etc.

NOTE 12
Depending on the implementation i.e. on the application requirements, the set of criteria might differ and lead to different protocol stacks.

14.4 Protocol stacks

14.4.1 OSI protocols

Full OSI stack: This scenario will map the CSTA service units on the OSI presentation layer. The definition of the OSI profile to be used is beyond the scope of this document.

Layer 1 to 3/4 stack: In these reduced stacks, the session, presentation and possibly transport layers are not used.

Limited scenario: In this scenario, the CSTA service units are directly mapped onto the data link layer.

The last two scenarios lead to some consequences which are further discussed below.

14.4.2 ISDN Interfaces

ISDN interfaces offer different types of scenarios that will be briefly discussed.

14.4.2.1 OSI stacks over ISDN bearer channels

The use of ISDN circuit or packet switched connection offer scenarios that do not differ from the OSI scenarios:

1. When a circuit switched connection is used, any of the scenarios defined elsewhere may be used over the B-channel bearer service.

2. When a packet switched connection based in CCITT Rec. X.25 is used, X.25 is retained at layer 2 and 3 of the bearer channel (either B or D-channel). When a full OSI stack is not used, or when non ISO protocols are used to provide the communication services, a set of additional functions, named above as convergence functions, will have to be defined and implemented. These can range from a simple mapping between service elements to the installation of a complete convergence protocol.

14.4.2.2 User-user signalling

When a user-user signalling bearer capability is defined and implemented over the ISDN D-channel, this capability could be used to support CSTA applications. However, the user must be conscious of the limitations provided by this capability such as no flow control, no segmentation, no error detection, no error recovery, no acknowledgement etc. In this particular case a Transport class 4 is most probably necessary.

In this connection-oriented context the Q.931 set of messages that can be used is reduced to connection control messages (SETUP, CONNECT, RELEASE, RELEASE COMPLETE etc.) and to one non acknowledged data transport mechanism (USER INFO message).

A possible mapping of the CSTA service primitives into the user-user signalling messages is shown in the following figure. This mapping assumes that ACSE is used to manage application associations and that ROSE is used to support the data transfer mode.

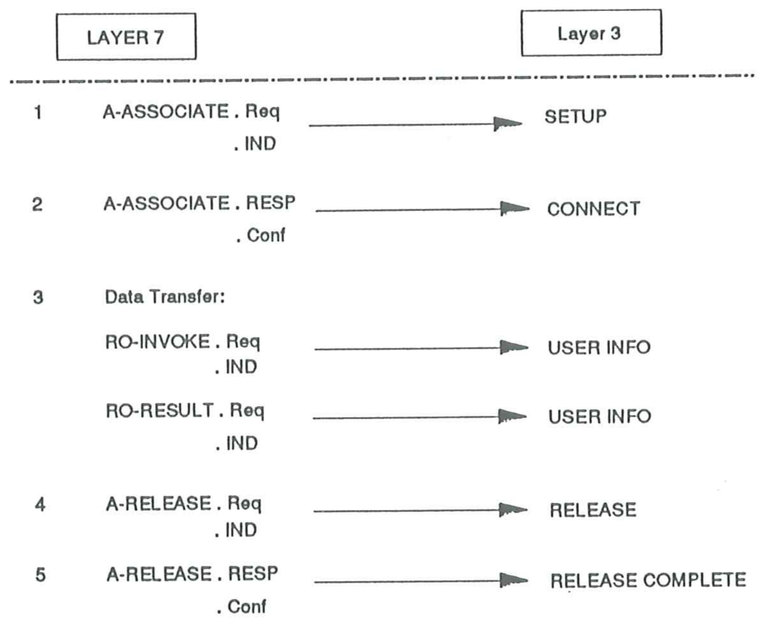


Figure 88 - User to User Signalling

The sequence shown in the figure is as follows:

1. A request to generate an association uses the A-ASSOCIATE.req primitive. This is mapped by ACSE into an APDU and uses the Rec. Q.931 SETUP message for transmission. When it arrives at the peer CSTA Application Entity AE, the APDU is extracted from the SETUP message and mapped by ACSE into the A-ASSOCIATE.ind primitive.

2. The receiving CSTA AE responds to the association request via the A-ASSOCIATE.resp primitive. This is mapped by ACSE into an APDU and uses the Rec. Q.931 CONNECT message for transmission. When it arrives at the initiating CSTA AE; the APDU is extracted from the CONNECT and mapped by ACSE into the A-ASSOCIATE.conf primitive.

3. Once the association is established, data transfer takes place using ROSE. This maps the various service interface primitives, such as RO-INVOKE, RO-RESULT, into the ROSE APDU. This APDU is carried between the peer CSTA AEs by the Rec. Q.931 USER INFO message.

4. When the association is no longer required, the A-RELEASE.req primitive is issued and mapped by ACSE into an APDU. This is transferred using the Rec. Q:931 RELEASE message and results on reception in an A-RELEASE.ind primitive.

5. The receiving CSTA AE responds to the association termination request via the A- RELEASE.resp primitive. This is mapped by ACSE into an APDU and uses the Rec. Q.931 RELEASE COMPLETE message for transmission. When it arrives at the CSTA AE,the APDU is extracted from the RELEASE COMPLETE and mapped by ACSE into the A-RELEASE.conf primitive.

14.4.2.3 Facility management (based on CCITT Rec.Q.932)

CCITT Rec. Q.932 proposes a functional architecture that allows facilities to be managed by use of a set of Rec. Q.931 messages. These messages provide connection management (REGISTER and RELEASE COMPLETE messages) as well as a data transfer mechanism (FACILITY message).

The mapping of the CSTA service primitives into these can also be envisaged with the same comments that have been made above regarding the data transfer mechanism limitations.

A possible mapping is depicted below.

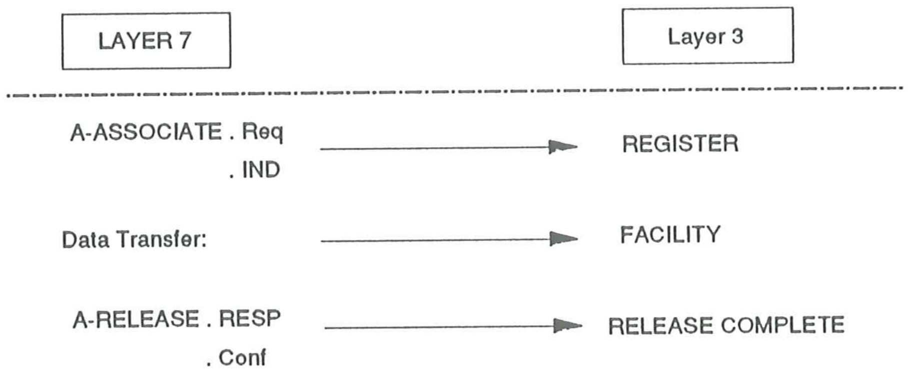


Figure 89 - Using Facility Information Elements

14.4.2.4 Signalling System 7 Stack

The use of a stack derived from the CCITT Signalling System 7 is also possible. In this context the mapping of the CSTA interconnection services onto the Signalling System 7 SCCP services is more appropriate. The SCCP provides the option of either a Connection Oriented Network Service or Connectionless Network Service. In both cases, the appropriate convergence functions will have to be used.

14.4.2.5 Other stacks

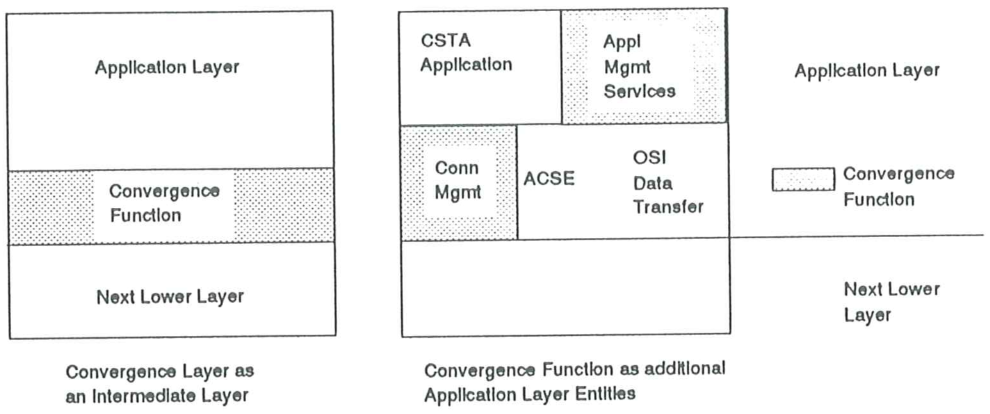


Figure 90 - Other Convergence Examples

Other stacks can be envisaged but the reservations expressed in the next clauses on the use of limited stacks will have to be carefully considered.

14.5 Consequences of the use of 'reduced' stacks

As described in 14.2, the use of 'reduced stacks' requires the introduction of convergence functions below the interconnection service boundary. Two cases have been presented in 14.2. We shall concentrate hereafter on the first case, where both the computing and the switching side are using the same protocol stacks.

It must be noted that the implementation of reduced stacks will lead to a service and quality of service which are degraded compared to the ones offered by a full OSI stack. As an example, the decision to implement a stack limited to the data link may be made under the following assumptions:

1. The link will always be point to point.
2. The data link provides a reliable service (with a low resulting bit error rate) and failures are always reported.
3. Link failures will be compensated by the implementation of management services at the application level.
4. Both computing and telecommunications systems will adopt the same date representation.
5. Message sizes will be limited due to the absence of segmentation mechanisms. Additional identifiers might have to be used at application level to allow parallel operations over the same data link.

14.6 Management of Communication Functions

14.6.1 Reliability/redundancy

As defined in clause 8.3, it will be an implementation option to provide redundant paths at the lower layers to guarantee a better reliability of the communications functions. Two basic mechanisms have been identified that are already available in the ISO toolkit:

1. HDLC multilink procedures as specified in Rec. X.75. These will offer load sharing capabilities over multiple physical point to point connections.
2. Provision of a complete OSI network service that will allow to route/reroute the traffic in case of link failures.

14.6.2 Congestion management - Recovery procedures

Depending on the implementation choice, a number of mechanisms normally provided by a full OSI stack may be missing. The absence of these mechanisms may affect the global QoS of the links and problems related to traffic overload or link failures may occur.

The architecture outlined above suggests that convergence functions will have to be implemented to provide a globally equivalent service. However, it should be understood that the concept of an abstract interconnection service boundary does not mean that this boundary must be implemented as a real interface.

In fact, convergence functions can be implemented:

1. either as an intermediate layer residing between the CSTA communication service boundary and the next lower layer (see previous figure);

OR

2. as an additional function residing within the CSTA application itself (see previous figure).

15. SECURITY AND MANAGEMENT IN CSTA

15.1 Security in CSTA

Security is acknowledged as being vital to CSTA.

The following material is a collection of preliminary information from discussions that have taken place up to now.

15.1.1 Overview

The purpose of security is to prevent unauthorised usage of resources within the CSTA domain. This is to ensure for example that privacy, charging and integrity are maintained.

Four levels of security have been identified:

1. Authentication of the identity of the Computing and Switching Functions.
2. Authentication of the application initiated at the client function by the server function.
3. Authentication of the user in the client function by the server function.
4. Access control which determines the scope of the user, client, application and client functions' authority to view and manipulate the specified objects.

Any combination of these security levels may be used to provide varying levels of security. However, all the levels of security in use must be satisfied prior to a CSTA application being successfully executed.

None of these security mechanisms should override existing security arrangements within the serving function.

15.1.2 Authentication and access control

ECMA TR/46 entitled "Security in Open Systems" provides a framework for the security proposals within this Technical Report. In particular, two aspects are considered:

1. Identity authentication
2. Access control

CSTA applications need to authenticate the identity of users, applications and functions.

The access control determines the objects that a valid user running a valid application from a valid sub-domain may have access to and the type of access permitted.

Authentication requires the use of secret information by both parties participating in the process. This may either be in the form of a password which is passed between them or by the use of some secret information which is not transferred, for example by using cryptographic key based schemes.

A cryptographic support service shall be available for secure data transfer, in particular for identity authentication. The decision not to use the encryption service shall be bi-laterally agreed by the Computing and Switching Functions.

Authentication shall be carried out at association time. In addition, a function shall be able to request further identity authentications of the user and application at any time. Authentication failure shall result in the association being broken.

Access control shall be done prior to accessing any CSTA object.

15.1.2.1 Identity Authentication

15.1.2.1.1 Objects and object groups

The authentication of objects which are within one function by the other function requires both to have knowledge of each other's objects.

The number of objects within a function is likely to be large, therefore a function may collect its objects into logical groups. These groups will have unique identities which are known to the other function. For example, the set of group 4 FAXs within the Switching Function may be given the collective name "FAX4". A CSTA request from the Computing Function to send a group 4 FAX will specify this name. Any of the FAX4 machines available within the Switching Function may be used. The FAX machine which is actually used and the time at which it is used, is determined by resource control mechanisms within the Switching Function.

The introduction of a new object within a function does not necessarily mean that the other function needs to be informed if the object can be included within an existing object group. For example, a new FAX4 machine can be installed in the Switching Function, within an existing object group, without informing the Computing Function. A subsequent request to use the FAX4 group may however result in a different scheduling mechanism being used.

The object groups and the access control lists may change. The manner in which this is achieved is the role of network management and is outside the scope of this Technical Report.

An object group may contain a single object.

Object groups may contain users, terminals, applications or other object groups.

An object group which contains users is called a user object group, similarly an object group of applications is called an application object group.

The identification of objects within the Computing Function is for further study.

15.1.2.1.2 Intra-Network Security

Users within a function may need to be identified to that function. Since this occurs wholly within one function it is not within the scope of CSTA. However, a function may need to supply a user identity to the other function.

15.1.2.1.3 Inter-network Security

At association time for an application, the Switching and Computing Function shall authenticate each other's identities.

The client function shall identify the user and the application to the server function. The user and application identity shall be specified in terms of their object groups. The server function shall either accept or reject the association. Rejection shall occur in one or more of the following situations:

1. Failure of the server to recognise the client function.
2. Failure of the client to recognise the server function.
3. Failure of the server function to recognise the server function application.
4. Failure of the server function to recognise the server function user.
5. Insufficient authority for the specified user to initiate the specified application on the server function.

The rejecting function may provide the reason for rejection.

15.1.2.2 Access Control

When a valid user in a valid client function, running a valid application wishes to access objects within the server function, access control determines the objects which may be assessed and the type of access permitted. The access control lists in the server function shall be determined upon the basis of the application object group and the user object group specified by the client function.

The access control lists indicate not only whether access to an object group is permissible, but also the type of access which is allowed.

15.1.3 Security Policy

All security mechanisms implemented wholly within a function are the responsibility of that function.

It is the responsibility of a function to perform security checks for itself rather than relying on security mechanisms within the other function.

Inter-function security mechanisms are decided by bi-lateral agreement between the two functions.

Any combination of the security functions available may be used to provide varying levels of security. None of the security functions are mandatory. None of them override or replace any existing security mechanisms within a function.

15.2 CSTA Management

CSTA management must be concerned with the following issues:

1. Load management
2. Control of the interaction between CSTA applications
3. Fault detection and management

The process of introducing new users, applications or objects is controlled by the network management function. The setting up, and amendment of access control lists is also the role of network management. These are therefore beyond the scope of this Technical Report, but may be needed for a CSTA application.

15.2.1 Charging Information

The identification of users which is an integral part of security may enable charging schemes to be developed within private networks. Charging within the public network is the subject of further study.

15.2.2 Load Management

CSTA will impose a new kind of load on both the switch and the computer. This extra load will be related to the number and type of the CSTA applications installed and satisfactory operation is dependent on a properly configured and sized system. The load can be considered in three categories:

1. Overall processing load can be related to the aggregate processing required in the computer and the switch.
2. Application load which is associated with a particular application.
3. Resource specific load which is related to a particular resource or piece of equipment.

Further study is required on the management of this load. Potential approaches include:

1. The server domain indicating its overall processing load to allow the client to modify its behaviour.
2. The server domain indicating that all further service requests from a particular application will be rejected until further notice. In this case, service requests which free resources may still be accepted.
3. The server domain rejecting individual service requests due to a lack of resources.

SECTION V

APPENDICES

APPENDIX A - REDUNDANCY

APPENDIX B - RELATING CSTA TO OSI LAYER 7 - ARCHITECTURE

APPENDIX C - VIRTUAL DEVICE CONCEPT

APPENDIX D - TRANSFER CONTEXT IN CSTA

APPENDIX E - OPEN DISTRIBUTED PROCESSING

APPENDIX A

REDUNDANCY

This Appendix is initial supporting material for the system reliability requirements stated in clause 7.

A.1 SCENARIOS

The decision to retain active redundancy or inactive redundancy must be made based on the requirements but should not reinvent basic principles unless it is shown as absolutely necessary.

1. Single association:

   In this case only one association exists between the two peer application entities. If redundancy is expected it can only be at link level. The multilink procedures should be transparent to the upper layers of the communication stack. These will provide a true load sharing mechanism. It is expected that recovery procedures are developed within the lower layers. It is not excluded however, that management procedures are used to get access to the status of the links.

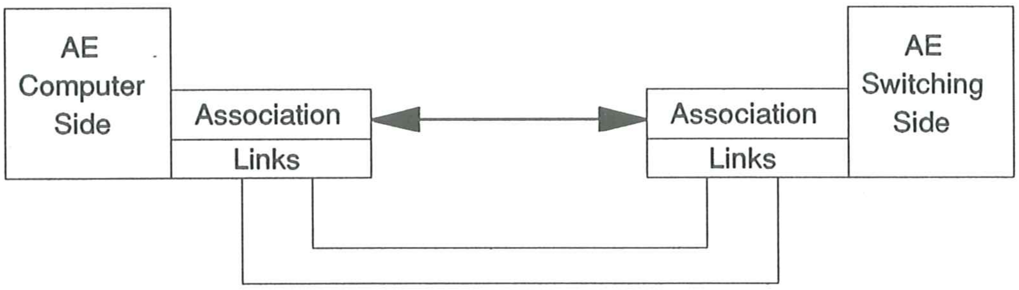


Figure A.1 - Single Association Topology

2. Duplicated AEs with hot back up:

   This scenario would imply the existence of two strictly identical application gateways supporting two AEs and working in load sharing mode (with hot backup procedures). Should one gateway crash, only the transient activities would be lost. The application process could be either monolithic or distributed. It will be in charge of maintaining an exact picture of the situation (established contexts, pending transactions) at any time.

   This scenario introduces a number of constraints and questions that have already been considered in other instances (e.g. Signalling System 7). It is felt that the solutions developed for Signalling System 7 (linksets, routesets) would be too costly to envisage in the CSTA context.

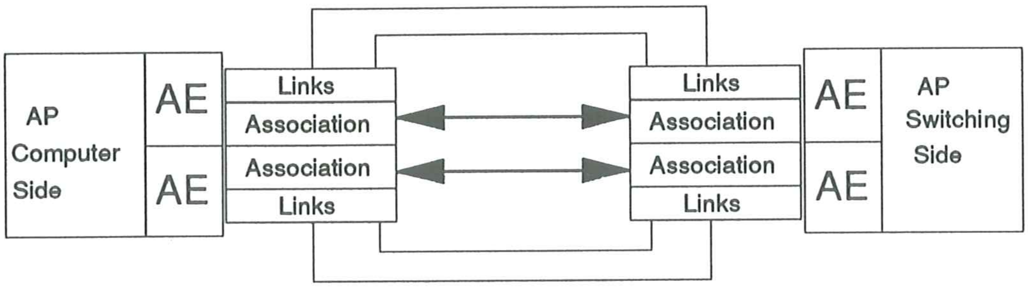


Figure A.2 - Duplicate Association Topology

3. Duplicated AEs and Standby mode of operation: In this scenario, the topology is similar to the one depicted above, except that only one gateway or association would be active at any time. The other associations, although established, would support no traffic (they would be inactive).

4. The ideal scenario would be fully dynamic:

   It would involve a communication subnetwork and would allow transactions initiated over one path to be concluded over another path.

   It must be noted that in scenarios 2 and 3, the AP or AP instances (when the AP is distributed) must have constantly a clear image of what goes on (contexts) at the gateway level so as to easily and smoothly recover from a gateway crash. Some agreements as to what these 'cold back up procedures' look like will have to be reached and standardized in order to reduce to a minimum the problems inherent to the transient (switchover) period.

   It is suggested that the management of the topology i.e. the control of the traffic over the possible routes is considered as a management function (configuration). The study of these functions could be associated with the studies on congestion that have been identified in previous pages.

A.2 INITIAL CONCLUSIONS

1. Considering active redundancy, out of the many possibilities, only two technical solutions seem to be viable:

   - Multilink solution: this assumes that the gateways are limited in functionality and operate as routers only

   - Warm failover context where each gateway maintains a communication context via an AE and an association. In this case, the correlation and orchestration of the activities of the two AEs belong to the AP.

     Complex recovery mechanisms and changeover procedures have to be developed at application level which might have to be standardized as well.

2. One simpler solution seems to be inactive redundancy whereby some AEs will sit totally idle until they are (re)activated by the CSTA AP, should the active gateway or communication channel crash. This solution pushes back the burden on the application in the hosts/switches where a smarter management process has to be put in place. But it presents the following advantages:

- It preserves the solution retained in the simple case by allowing to develop more complex scenarios as supersets of the simple ones.
- It is OSI compatible (only one association at a time).
- It pushes back the solution at the application level and therefore leaves greater flexibility to the implementers.
- It should allow to develop additional mechanisms in an almost independent way.

Negative aspects however, have to be recognised and accepted:

- The changeover procedures leave some downtime window. A downtime that would not exceed 10 seconds is probably acceptable (but needs to be defined).
- Agreements on the detection and recovery principles must be developed and standardized. Agreements on what is lost during a changeover must be reached and the corresponding set of error messages must be standardized.
- Possible collisions are not totally excluded.
- An additional set of messages have to be put in place that one could consider as part of network (or configuration) management.

APPENDIX B

RELATING CSTA TO OSI LAYER 7 - ARCHITECTURE

The purpose of this Appendix is to encourage discussion regarding OSI Layer 7 structure by relating concepts from these areas of the Technical Report:

- the Application Examples of Clause 6;
- the notions of Client and Server, Computing Function and Switching Function, and CSTA Service Boundary of Clause 8;
- the CSTA Service Descriptions of Clause 10;
- the ideas of Application Layer elements of Clause 13 to Layer 7 of the OSI model.

B.1 OSI APPLICATION PROCESS (AP)

This Appendix uses the example of a fictitious department store accounting program package to illustrate the position of the CSTA architecture in relationship to the architecture of OSI Layer 7. We speculate that the application package has an Accounts Receivable component, an Accounts Payable component, an Inventory component and others.

The Accounts Receivable portion might have a Database facility for keeping track of the customers who have charge accounts with that store, and Communications Facility for services to the store employees who use computers and terminals for store business that is heavily involved with the use of the telephone. The Communications Facility is an excellent example of an integrated voice and data application that could use CSTA, and it is the emphasis of this appendix. We suggest that this programming package would reside in an OSI Application Process. Further, the CSTA Services of Clause 10 would be the CSTA specific service elements.

B.1.1 CSTA Application Examples

The Communications Facility component might use telecommunications and data features as defined in these CSTA Application Examples, taken from Clause 6:

1. Telemarketing

   Services of this category could involve automatically calling customers to personally notify them of upcoming private sales.

2. Personal User Support

   These services would enhance the human usability of the store's telephony system by providing screen-based control of telephony functions, such as conference, transfer, hold etc.

3. Customer Support Environment

   These services could augment Personal User Services to allow the referral of a transaction to a specialist agent. The transfer would be integrated, meaning that the voice call and the data screen would be transferred as a single entity.

   These application classes would execute Telecommunications related Application Service Primitives for CSTA Services.

Similarly, there could be applications operating in the Switching Function that use services that are in the Computing Function. Some examples include Messaging Services, ACD Agent Statistics and Telemarketing.

B.2 OSI LAYER 7

B.2.1 CSTA-Specific Service Elements

This is where the majority of CSTA would reside. Here the Telecommunications related service primitives are interpreted, checked for validity, and mapped to CSTA Application Protocol Data Units. The CSTA Application Protocol Data Units are then transported to the remote partner.

B.2.1.1 CSTA Application Protocol Data Units

Protocol Data Units are sent between the client CSTA-Specific ASE and the corresponding server CSTA-Specific ASE.

B.2.2 Common Service Elements

Other Common Service Elements could be accessible by the CSTA Specific ASEs. These could include:

1. Association Control Service Element (ACSE).
2. Remote Operations Service Element (ROSE).
3. Transaction Processing Applications Service Element (TP/ASE).

The following Figure shows that the CSTA Specific ASEs are one among many OSI Layer 7 Service Elements, and how these concepts map to the 7-layer OSI model.

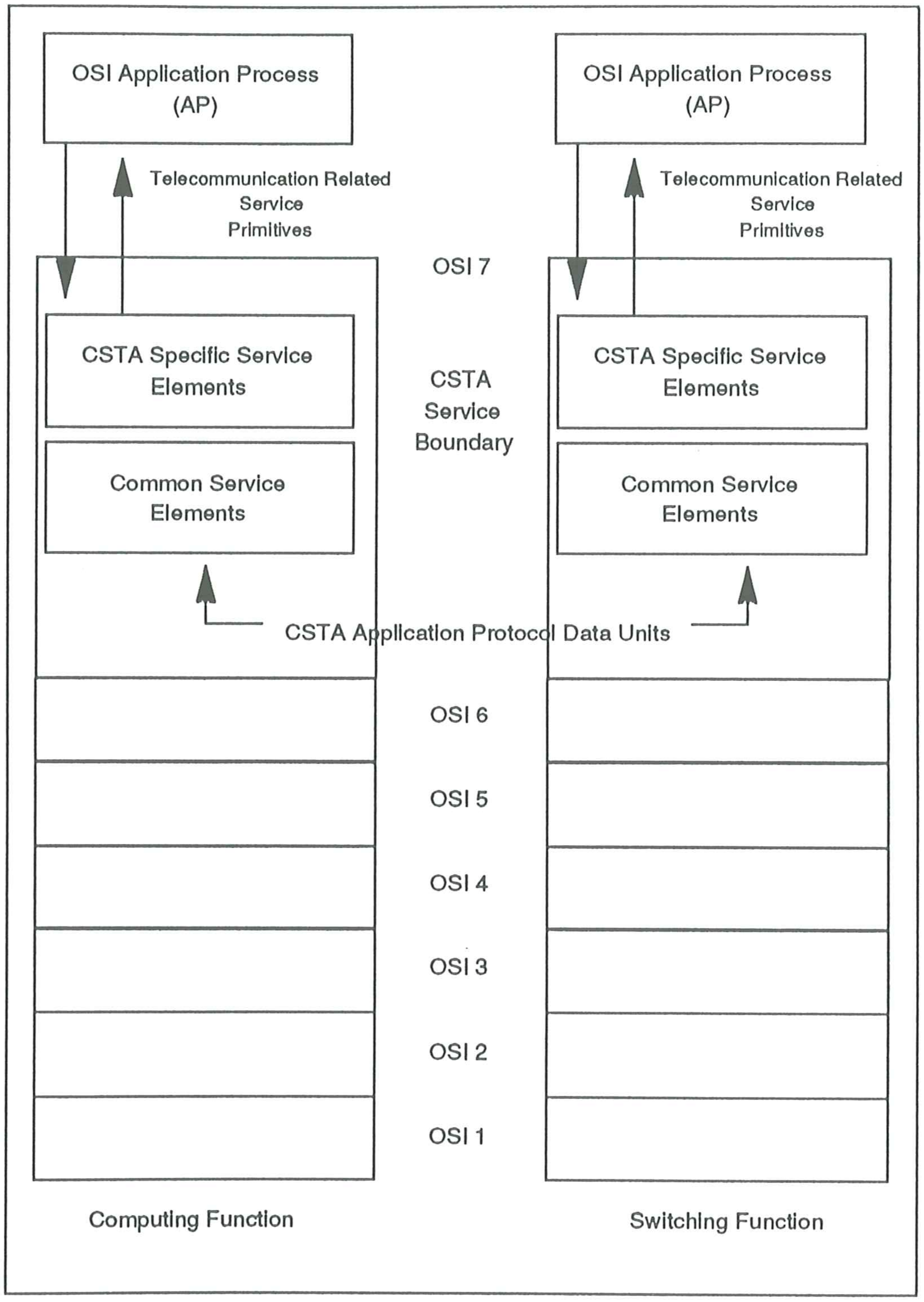

OSI Application Process (AP)
OSI Application Process (AP)
Telecommunication Related Service Primitives
Telecommunication Related Service Primitives
OSI 7
CSTA Specific Service Elements
CSTA Specific Service Elements
CSTA Service Boundary
Common Service Elements
Common Service Elements
CSTA Application Protocol Data Units
OSI 6
OSI 5
OSI 4
OSI 3
OSI 2
OSI 1
Computing Function
Switching Function

Figure B.1 - Distribution of CSTA Driver ASEs

APPENDIX C

VIRTUAL DEVICE CONCEPT

There may be situations in which an application wishes to establish a telephone call even though it has so far determined only one of the two devices that are to engage in the call. 'Predictive dialling' and 'Telecommuting' are examples of applications in which this may be the case. ('Examples of Functions using Virtual Device Configurations' on the following page illustrates these applications.)

Predictive dialling is a sophisticated type of outbound telemarketing where an application places calls on behalf of some number of sales agents. The application may request that a call be placed to a prospective customer, or even to multiple customers, at a time when all sales agents are already involved in calls. The application is predicting that some number of the outbound calls will be unsuccessful, and that by the time one of the calls to a potential customer is successfully established, one of the busy agents will have become idle and can handle the sales call.

In the telecommuting application, the agent performs his work from home. A single telephone connection is established to the agent when he logs into the system, and it is used throughout the day for servicing individual customers.

To facilitate such environments, the concept of a "virtual device", that is, a device not associated with an actual device or person, is introduced. In the predictive dialling example, the application could request that a telephone call be established between two devices, one of which is a "real" device, i.e. the potential customer, while the other is a "virtual" device. If the "real" device answers the call, the application must be ready to quickly insert a second "real" device into the call.

Virtual devices are treated no differently than "real" devices. Any service that an application can request on behalf of a "real" device can be requested on behalf of a virtual device. For example, an application can request that a virtual device be added to a call, that a virtual device be transferred out of a call, that a call be redirected from a virtual device, and so on.

The number of virtual devices that can be provided by the network for this capability is a network option.

C.1 EXAMPLES OF FUNCTIONS USING VIRTUAL DEVICE CONFIGURATIONS

The concept of a Make_Call in which one of the devices may be a virtual device was introduced to support functions such as predictive dialling and telecommuting. It is possible to perform these functions with real devices, but virtual device is an optional optimization that might be interesting to switch manufacturers. The idea of a "partial call", as represented by a call in which one of the two devices is a virtual device, was required. Figures C.1 and C.2 show examples of the use of virtual devices to achieve these functions.

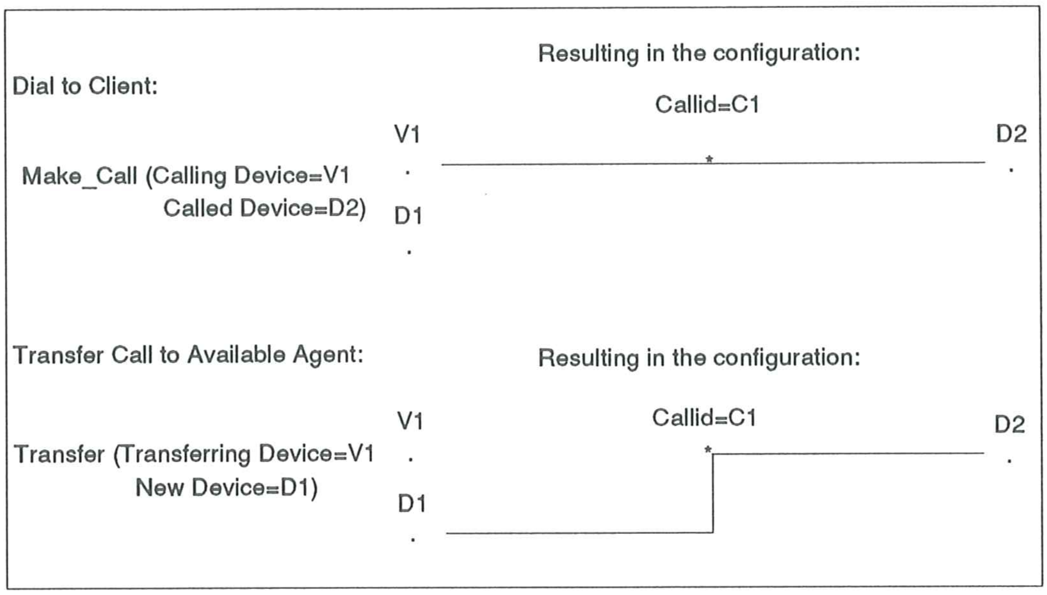

Dial to Client:
Make_Call (Calling Device=V1
Called Device=D2)
Resulting in the configuration:
Callid=C1
V1
D1
D2
*
Transfer Call to Available Agent:
Transfer (Transferring Device=V1
New Device=D1)
Resulting in the configuration:
Callid=C1
V1
D1
D2
*

Figure C.1 - Predictive Dialing Scenario

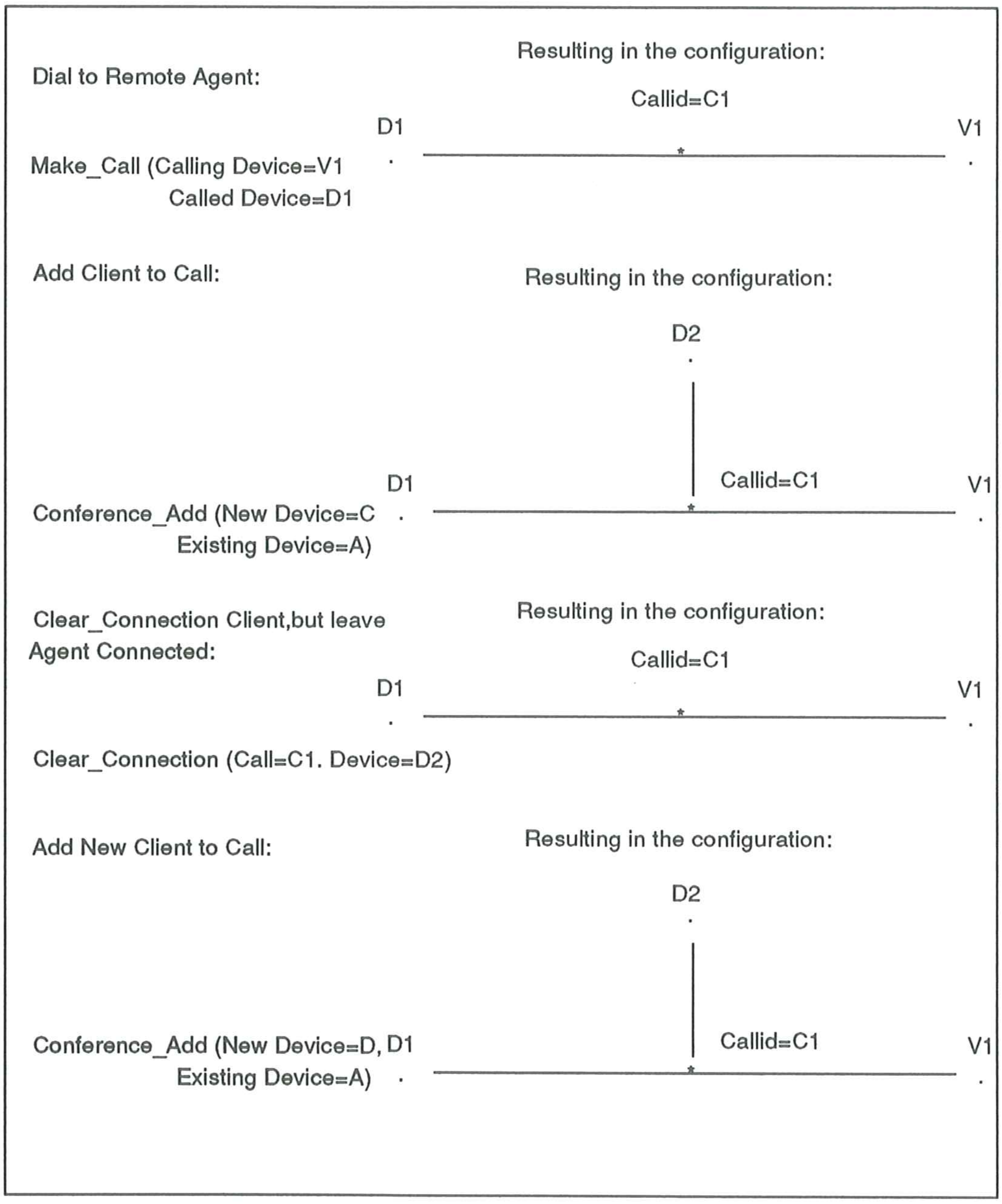

Dial to Remote Agent:
Resulting in the configuration:
Callid=C1
D1
V1
Make_Call (Calling Device=V1
Called Device=D1
Add Client to Call:
Resulting in the configuration:
D2
D1
Callid=C1
V1
Conference_Add (New Device=C
Existing Device=A)
Clear_Connection Client,but leave Agent Connected:
Resulting in the configuration:
Callid=C1
D1
V1
Clear_Connection (Call=C1. Device=D2)
Add New Client to Call:
Resulting in the configuration:
D2
Conference_Add (New Device=D,
Existing Device=A)
D1
Callid=C1
V1

Figure C.2 - Telecommuting Scenario

The Telecommuting application requires that the connection to the agent not be released when the customer finishes the transaction and hangs up.

Figure C.2 shows how the virtual device concept can be used to support this application. The telecommuting agent is at D1. The agent's connection to the system is shown as a call to the virtual device V1. When a customer call is assigned to that agent, the customer device D3 is added to the two-device call consisting of device D1 and virtual device V1, device V1 remains in the call. When the transaction is completed, the customer is removed from the call (either by manually hanging up or by CSTA action); the virtual device still remains in the call with the agent at D1, so the connection to D1 is not automatically released. An example enlarging on the material of clause 13 is given in Appendix B.

APPENDIX D

TRANSFER CONTEXT IN CSTA

There is a Transfer__Call service in CSTA which allows an application in the computing domain to transfer a call from a terminal in the switching domain to another terminal in that domain. An application running within the computing domain may also transfer an application context in the computing domain which is associated with a particular computer terminal to another terminal. In this way a user who has a switching domain voice terminal and a computer terminal may transfer both his voice call (switching domain) and application context (computing domain) to another user.

This Appendix details a Transfer service which can be initiated from the switching domain. If a user has a voice call established and information displayed on his computer terminal, when he transfers his voice call, the information on his computer terminal and the state of his application process will be transferred also. That is, in the computing domain the application context is transferred. For this reason the service has been called the Transfer__Context service.

The Transfer__Context service allows the Switching Function to transfer the context of an application which is associated with a user terminal in the computing domain to another terminal in that domain.

D.1 APPLICATION EXAMPLE

In a telemarketing application an agent receives a telephony call from a customer on device S1 in the switching domain. He calls up information relating to this customer which is displayed on his terminal C1 in the computing domain.

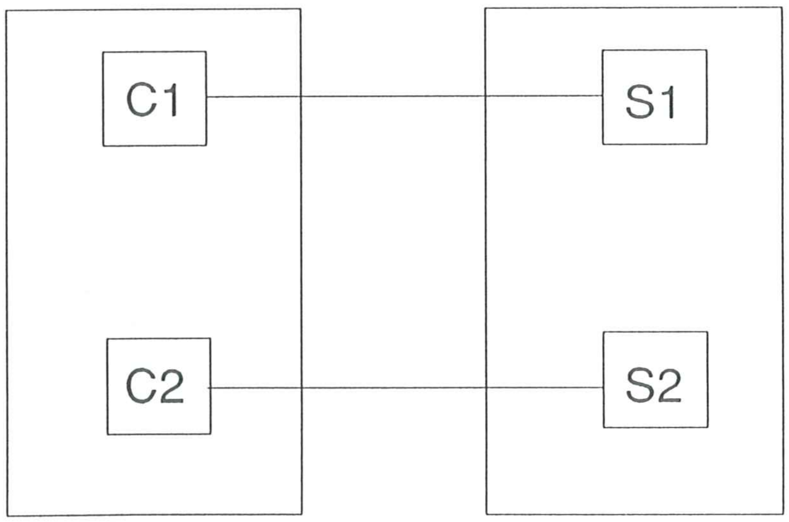


Figure D.1 - Application Example

When he transfers his call from S1 to another agent's voice terminal, S2 in the switching domain a Transfer__Context service request is sent from the Switching Domain to the Computing Domain which results in the context of the computing application being transferred from C1 to C2. Therefore the information which is available on the first agent screen is available to the second agent, who may then continue running the application initiated by this agent.

APPENDIX E

OPEN DISTRIBUTED PROCESSING

There is widespread interest in the topic of Open Distributed Processing (ODP). ECMA has conducted some preliminary work in this area resulting in Technical Report TR/49, "Support Environment for Open Distributed Processing (SD-ODP)".

This Appendix considers the relationship of TR/49, its subject matter and subsequent related work to the development of the CSTA architecture.

E.1 SE-ODP

Although the following text attempts to informally summarize ECMA TR/49 on SE-ODP, it is no substitute for a full evaluation of the complete document itself (this summary is also not rigorous in its use of terms)

E.1.1 Modular software and distributed processing

SE-ODP is concerned with supporting modular software and distributed processing:

1. Modularity

   Designing applications that are formed from program modules whose contents are encapsulated behind defined interfaces. So the key concepts are modules + interfaces + encapsulation.

2. Distribution

   The individual modules execute in independent, autonomous environments (so for example, equipment failure may only directly affect a single module). The effects of distribution, or separation, may be observable in the way the modules interact (see later text on distribution transparency). So the key concepts are autonomy + separation.

3. Wide applicability

   The system must be dependable (which covers reliability, availability, performance, security and safety) and adaptable (i.e. scaleable) for small and large systems. So the key concepts are dependability + scaling.

4. Reusability

   The system modules must be capable of reuse and relocation. This implies the key concept of portability.

5. Open systems

   The distributed system should be open to different types of equipment, operating systems etc. Management of the distributed system should not require a single, centralised management authority so that the individual modules can co-exist with federated control. So the key concepts are heterogeneity + federation.

E.1.2 Viewpoints

Open Distributed Processing attempts to partition the problems of distributed processing by introducing a framework of abstractions where every system can be perceived from different viewpoints. The same system is modelled from each of the following viewpoints:

1. The enterprise viewpoint which is concerned with the role of the system within the business organisation. It describes the system in terms of people and actions and has a similar perspective to the applications described in clause 6.

2. The information viewpoint which is concerned with the location and processing of information. It describes information in terms of such things as its structure, interpretation, flow and manipulation etc.

3. The computation viewpoint which describes the system in terms of the operations and algorithms performed. It is concerned with structuring the applications so that they are independent of the underlying computers, switches and networks. It is a level of abstraction which is concerned with what is done, not how it is done.

4. The engineering viewpoint which describes the engineering necessary to support the system. It is concerned with providing desired characteristics such as performance, dependability and distribution transparency. It is therefore more concerned with how things are done, rather than what things are done.

5. The technology viewpoint which is concerned with the components (hardware and software) that are used to build the system.

Certain kinds of pervasive concerns (such as security) are classified as "aspects", and are modelled in several of the viewpoints.

ECMA TR/49 on SE-ODP is primarily concerned with the computation and engineering viewpoints.

E.1.3 Distribution Transparency

A central consideration of the SE-ODP is the identification and support of various Distribution Transparencies for hiding the consequences of distribution:

1. access transparency: which hides the mechanisms used to achieve interaction;

2. location transparency: which hides the location of components;

3. concurrency transparency: which hides simultaneous access to shared resources;

4. replication transparency: which hides replication of a component into multiple separate instances;

5. failure transparency: which hides the effects of component failure;

6. migration transparency: which hides the effects of relocating a component.

The SE-ODP has indentified the above points as key characteristics of distributed processing and has developed a model for supporting selective use of these distribution transparencies. In particular, the computing viewpoint defines the required distribution transparencies; the engineering viewpoint describes how the desired distribution transparencies are provided.

E.1.4 Object modelling

SE-ODP adopts an object modelling technique where ODP processing components (which are akin to the modules mentioned earlier) interact via ODP interfaces. The specification of an ODP interface defines all the valid interactions that can occur between the associated ODP processing components. The object models are represented by object diagrams with a defined notation. Different objects are visible for each of the viewpoints reflecting the various degrees of focus. In particular:

1. The computational viewpoint is represented by objects describing the various distributed algorithmic functions and the interactions between these functions. The precise level of granularity is left to the system designer.

2. The engineering viewpoint introduces native components (which are implementation dependent) and interface adaptors which provide the interface between the ODP component and the native components (i.e. they map from standardized interactions to non-standard interactions).

E.1.5 Interaction structure

A common model of interaction semantics is required for all ODP interfaces. Existing computational models are based on non-concurrent, single-processor environments and do not adequately cope with various aspects associated with distribution of modular systems (e.g. encapsulation, concurrency, distribution transparency, atomicity, etc). Once a common semantic model has been defined, it is necessary to specify interactions via one, or more, "interface definition languages" (of course, each must conform to the general semantic model).

In the computational viewpoint, the interface definition notation specifies an abstract interface between processing components that is independent of any particular choice of concrete form. Since this specification covers both the semantics and syntax of the abstract interface it is described via a "language" (semantics and syntax) rather than just a "notation" (syntax).

In the engineering viewpoint, this abstract representation is converted to an Application Programming Interface which has a concrete form that is necessarily committed to some particular syntax, control structure and language binding. There may be multiple concrete interfaces representing the same abstract interface described by the interface definition language in the computation viewpoint. Each interaction also has a transfer syntax (which defines the representation during transit) visible from the engineering viewpoint. Different transfer syntaxes may be applicable depending on the precise details of the distribution.

E.1.6 Configuration

SE-ODP is also concerned with the configuration of distributed systems, types of client/server or producer/consumer models and the necessary associated bindings (i.e. how a client locates an appropriate server).

E.2 CSTA AND SE-ODP

Comparisons can be made with the concepts in SE-ODP and the contents in this Technical Report on CSTA:

1. Both are concerned with modular, distributed systems.
2. Both advocate a phased approach to system design such that the overall problem can be broken down into smaller steps:
   - the ODP approach: where each viewpoint is concerned with a different part of the problem;
   - the CCITT three stage process: which also breaks the design process down into successive steps with each stage focusing on providing supporting detail for the earlier stages.

   In particular, the service descriptions contained in Clause 10 of this Technical Report are similar in levels of abstraction, but not of course notation, to those identified for the abstract ODP interfaces.
3. CSTA and SE-ODP are both concerned primarily with the computational and engineering viewpoint.
4. Both are concerned with the selective provision of distribution transparencies.
5. Both adopt a form of object modelling to describe the distributed system.

6. Both are concerned with client/server type relationships and the associated problems of naming, binding etc.

E.3 SUMMARY

There is no agreement that SE-ODP is appropriate to CSTA. However, it appears that SE-ODP could influence the further development of the CSTA architecture. Both topics are, of course, subject to further work which will be conducted in parallel. Any potential benefits depend on early development of the key SE-ODP components, which include:

- definition of the common semantic model to support distributed interactions;
- development of appropriate configuration and binding mechanisms between the distributed components;
- specification of a suitable interface definition language.

One potential concern is that CSTA, unlike many applications using SE-ODP, is expected to have stringent real-time requirements.